



OPTIX API REFERENCE

v4.0

API Reference Manual



1 Module Documentation

1.1 OpenGL Texture Formats

The following OpenGL texture formats are available for interoperability with OptiX.

| |
|----------|
| R8I |
| R8UI |
| RG8I |
| RG8UI |
| RGBA8 |
| RGBA8I |
| RGBA8UI |
| R16I |
| R16UI |
| RG16I |
| RG16UI |
| RGBA16 |
| RGBA16I |
| RGBA16UI |
| R32I |
| R32UI |
| RG32I |
| RG32UI |
| RGBA32I |
| RGBA32UI |
| R32F |
| RG32F |
| RGBA32F |

1.2 DXGI Texture Formats

The following DXGI texture formats are available for interoperability with OptiX.

| |
|--------------------|
| R8_SINT |
| R8_SNORM |
| R8_UINT |
| R8_UNORM |
| R16_SINT |
| R16_SNORM |
| R16_UINT |
| R16_UNORM |
| R32_SINT |
| R32_UINT |
| R32_FLOAT |
| R8G8_SINT |
| R8G8_SNORM |
| R8G8_UINT |
| R8G8_UNORM |
| R16G16_SINT |
| R16G16_SNORM |
| R16G16_UINT |
| R16G16_UNORM |
| R32G32_SINT |
| R32G32_UINT |
| R32G32_FLOAT |
| R8G8B8A8_SINT |
| R8G8B8A8_SNORM |
| R8G8B8A8_UINT |
| R8G8B8A8_UNORM |
| R16G16B16A16_SINT |
| R16G16B16A16_SNORM |
| R16G16B16A16_UINT |
| R16G16B16A16_UNORM |
| R32G32B32A32_SINT |
| R32G32B32A32_UINT |
| R32G32B32A32_FLOAT |

2 Class Documentation

2.1 optix::Aabb Class Reference

2.1.1 Detailed Description

Axis-aligned bounding box.

Description

[Aabb](#) is a utility class for computing and manipulating axis-aligned bounding boxes (aabb). [Aabb](#) is primarily useful in the bounding box program associated with geometry objects. [Aabb](#) may also be useful in other computation and can be used in both host and device code.

History

[Aabb](#) was introduced in OptiX 1.0.

See also [RT_PROGRAM](#), [rtGeometrySetBoundingBoxProgram](#)

Public Member Functions

- RT_HOSTDEVICE [Aabb](#) ()
- RT_HOSTDEVICE [Aabb](#) (const float3 &min, const float3 &max)
- RT_HOSTDEVICE [Aabb](#) (const float3 &v0, const float3 &v1, const float3 &v2)
- RT_HOSTDEVICE bool [operator==](#) (const [Aabb](#) &other) const
- RT_HOSTDEVICE float3 & [operator\[\]](#) (int i)
- RT_HOSTDEVICE const float3 & [operator\[\]](#) (int i) const
- RT_HOSTDEVICE void [set](#) (const float3 &min, const float3 &max)
- RT_HOSTDEVICE void [set](#) (const float3 &v0, const float3 &v1, const float3 &v2)
- RT_HOSTDEVICE void [invalidate](#) ()
- RT_HOSTDEVICE bool [valid](#) () const
- RT_HOSTDEVICE bool [contains](#) (const float3 &p) const
- RT_HOSTDEVICE bool [contains](#) (const [Aabb](#) &bb) const
- RT_HOSTDEVICE void [include](#) (const float3 &p)
- RT_HOSTDEVICE void [include](#) (const [Aabb](#) &other)
- RT_HOSTDEVICE void [include](#) (const float3 &min, const float3 &max)
- RT_HOSTDEVICE float3 [center](#) () const
- RT_HOSTDEVICE float [center](#) (int dim) const
- RT_HOSTDEVICE float3 [extent](#) () const
- RT_HOSTDEVICE float [extent](#) (int dim) const
- RT_HOSTDEVICE float [volume](#) () const
- RT_HOSTDEVICE float [area](#) () const
- RT_HOSTDEVICE float [halfArea](#) () const
- RT_HOSTDEVICE int [longestAxis](#) () const
- RT_HOSTDEVICE float [maxExtent](#) () const
- RT_HOSTDEVICE bool [intersects](#) (const [Aabb](#) &other) const
- RT_HOSTDEVICE void [intersection](#) (const [Aabb](#) &other)
- RT_HOSTDEVICE void [enlarge](#) (float amount)
- RT_HOSTDEVICE bool [isFlat](#) () const
- RT_HOSTDEVICE float [distance](#) (const float3 &x) const
- RT_HOSTDEVICE float [distance2](#) (const float3 &x) const
- RT_HOSTDEVICE float [signedDistance](#) (const float3 &x) const

Public Attributes

- float3 [m_min](#)
- float3 [m_max](#)

2.1.2 Constructor & Destructor Documentation

2.1.2.1 OPTIXU_INLINE RT_HOSTDEVICE optix::Aabb::Aabb ()

Construct an invalid box

2.1.2.2 OPTIXU_INLINE RT_HOSTDEVICE optix::Aabb::Aabb (const float3 & *min*, const float3 & *max*)

Construct from min and max vectors

2.1.2.3 OPTIXU_INLINE RT_HOSTDEVICE optix::Aabb::Aabb (const float3 & *v0*, const float3 & *v1*, const float3 & *v2*)

Construct from three points (e.g. triangle)

2.1.3 Member Function Documentation

2.1.3.1 OPTIXU_INLINE RT_HOSTDEVICE float optix::Aabb::area () const

Compute the surface area of the box

2.1.3.2 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::Aabb::center () const

Compute the box center

2.1.3.3 OPTIXU_INLINE RT_HOSTDEVICE float optix::Aabb::center (int *dim*) const

Compute the box center in the given dimension

2.1.3.4 OPTIXU_INLINE RT_HOSTDEVICE bool optix::Aabb::contains (const float3 & *p*) const

Check if the point is in the box

2.1.3.5 OPTIXU_INLINE RT_HOSTDEVICE bool optix::Aabb::contains (const Aabb & *bb*) const

Check if the box is fully contained in the box

2.1.3.6 OPTIXU_INLINE RT_HOSTDEVICE float optix::Aabb::distance (const float3 & *x*) const

Compute the minimum Euclidean distance from a point on the surface of this [Aabb](#) to the point of interest

2.1.3.7 OPTIXU_INLINE RT_HOSTDEVICE float optix::Aabb::distance2 (const float3 & *x*) const

Compute the minimum squared Euclidean distance from a point on the surface of this [Aabb](#) to the point of interest

2.1.3.8 OPTIXU_INLINE RT_HOSTDEVICE void optix::Aabb::enlarge (float *amount*)

Enlarge the box by moving both min and max by 'amount'

2.1.3.9 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::Aabb::extent () const

Compute the box extent

2.1.3.10 OPTIXU_INLINE RT_HOSTDEVICE float optix::Aabb::extent (int *dim*) const

Compute the box extent in the given dimension

2.1.3.11 OPTIXU_INLINE RT_HOSTDEVICE float optix::Aabb::halfArea () const

Compute half the surface area of the box

2.1.3.12 OPTIXU_INLINE RT_HOSTDEVICE void optix::Aabb::include (const float3 & *p*)

Extend the box to include the given point

2.1.3.13 OPTIXU_INLINE RT_HOSTDEVICE void optix::Aabb::include (const Aabb & *other*)

Extend the box to include the given box

2.1.3.14 OPTIXU_INLINE RT_HOSTDEVICE void optix::Aabb::include (const float3 & *min*, const float3 & *max*)

Extend the box to include the given box

2.1.3.15 OPTIXU_INLINE RT_HOSTDEVICE void optix::Aabb::intersection (const Aabb & *other*)

Make the current box be the intersection between this one and another one

2.1.3.16 OPTIXU_INLINE RT_HOSTDEVICE bool optix::Aabb::intersects (const Aabb & *other*) const

Check for intersection with another box

2.1.3.17 OPTIXU_INLINE RT_HOSTDEVICE void optix::Aabb::invalidate ()

Invalidate the box

2.1.3.18 OPTIXU_INLINE RT_HOSTDEVICE bool optix::Aabb::isFlat () const

Check if the box is flat in at least one dimension

2.1.3.19 OPTIXU_INLINE RT_HOSTDEVICE int optix::Aabb::longestAxis () const

Get the index of the longest axis

2.1.3.20 OPTIXU_INLINE RT_HOSTDEVICE float optix::Aabb::maxExtent () const

Get the extent of the longest axis

2.1.3.21 OPTIXU_INLINE RT_HOSTDEVICE bool optix::Aabb::operator== (const Aabb & *other*) const

Exact equality

2.1.3.22 OPTIXU_INLINE RT_HOSTDEVICE float3 & optix::Aabb::operator[] (int *i*)

Array access

2.1.3.23 OPTIXU_INLINE RT_HOSTDEVICE const float3 & optix::Aabb::operator[] (int *i*) const

Const array access

2.1.3.24 OPTIXU_INLINE RT_HOSTDEVICE void optix::Aabb::set (const float3 & *min*, const float3 & *max*)

Set using two vectors

2.1.3.25 OPTIXU_INLINE RT_HOSTDEVICE void optix::Aabb::set (const float3 & *v0*, const float3 & *v1*, const float3 & *v2*)

Set using three points (e.g. triangle)

2.1.3.26 OPTIXU_INLINE RT_HOSTDEVICE float optix::Aabb::signedDistance (const float3 & *x*) const

Compute the minimum Euclidean distance from a point on the surface of this [Aabb](#) to the point of interest. If the point of interest lies inside this [Aabb](#), the result is negative

2.1.3.27 OPTIXU_INLINE RT_HOSTDEVICE bool optix::Aabb::valid () const

Check if the box is valid

2.1.3.28 OPTIXU_INLINE RT_HOSTDEVICE float optix::Aabb::volume () const

Compute the volume of the box

2.1.4 Member Data Documentation

2.1.4.1 float3 optix::Aabb::m_max

Max bound

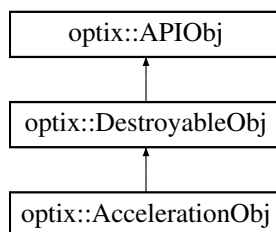
2.1.4.2 float3 optix::Aabb::m_min

Min bound

2.2 optix::AccelerationObj Class Reference

2.2.1 Detailed Description

Acceleration wraps the OptiX C API RTacceleration opaque type and its associated function set. Inheritance diagram for optix::AccelerationObj:



Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context](#) [getContext](#) () const
- [RTacceleration](#) [get](#) ()

- void [markDirty](#) ()
- bool [isDirty](#) () const

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

- RTsize [getDataSize](#) () const
- void [getData](#) (void *data) const
- void [setData](#) (const void *data, RTsize size)

Friends

- class **Handle**< **AccelerationObj** >

Additional Inherited Members

2.2.2 Member Function Documentation

2.2.2.1 `RTsize optix::AccelerationObj::getDataSize () const` `[inline]`

Deprecated in OptiX 4.0 Query the size of the marshalled acceleration data. See [rtAccelerationGetDataSize](#).

2.2.2.2 `std::string optix::AccelerationObj::getProperty (const std::string & name) const` `[inline]`

Query properties specifying Acceleration builder behavior. See [rtAccelerationGetProperty](#).

2.2.2.3 `void optix::AccelerationObj::markDirty ()` `[inline]`

Mark the acceleration as needing a rebuild. See [rtAccelerationMarkDirty](#).

2.2.2.4 `void optix::AccelerationObj::setProperty (const std::string & name, const std::string & value)` `[inline]`

Set properties specifying Acceleration builder behavior. See [rtAccelerationSetProperty](#).

2.3 optix::APIObj Class Reference

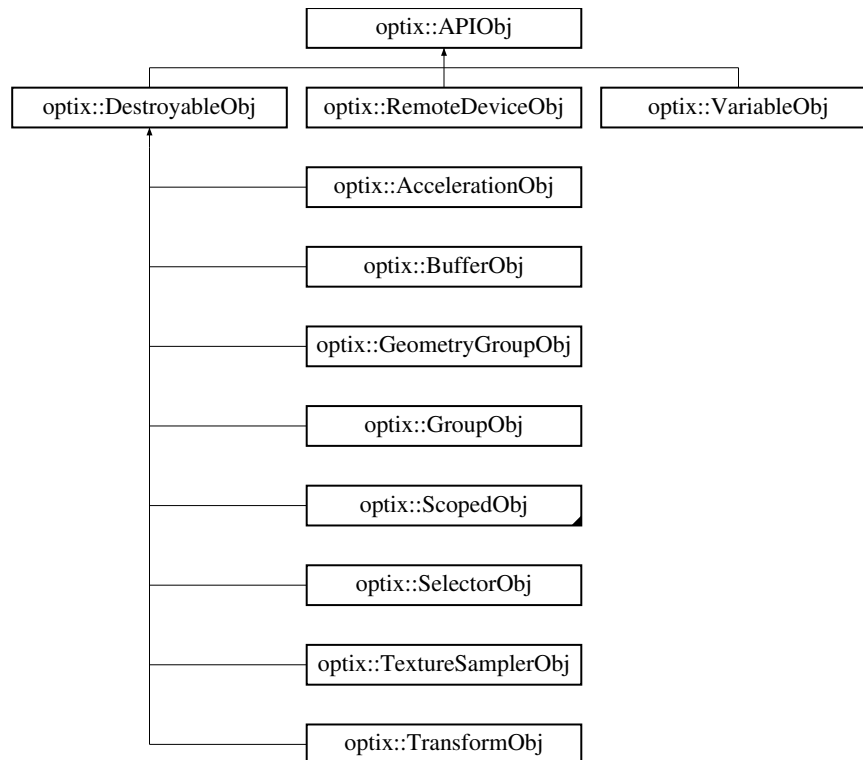
2.3.1 Detailed Description

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

Inheritance diagram for optix::APIObj:



Public Member Functions

- void [addReference](#) ()
- int [removeReference](#) ()
- virtual [Context](#) [getContext](#) () const =0
- virtual void [checkError](#) ([RResult](#) code) const
- virtual void **checkError** ([RResult](#) code, [Context](#) context) const
- void **checkErrorNoGetContext** ([RResult](#) code) const

Static Public Member Functions

- static [Exception](#) [makeException](#) ([RResult](#) code, [RTcontext](#) context)

2.3.2 Member Function Documentation

2.3.2.1 void optix::APIObj::checkError ([RResult](#) code) const [inline],[virtual]

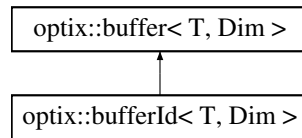
Check the given result code and throw an error with appropriate message if the code is not RTsuccess

Reimplemented in [optix::ContextObj](#).

2.4 optix::boundCallableProgramId< T > Class Template Reference

2.5 optix::buffer< T, Dim > Struct Template Reference

Inheritance diagram for optix::buffer< T, Dim >:



Classes

- struct [type](#)

Public Types

- typedef [VectorTypes](#)< size_t, Dim > **WrapperType**
- typedef [VectorTypes](#)< size_t, Dim >::Type **IndexType**

Public Member Functions

- `__device__ __forceinline__ IndexType size () const`
- `__device__ __forceinline__ T & operator[] (IndexType i)`

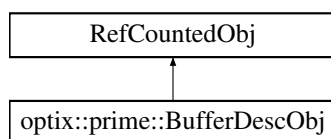
Static Protected Member Functions

- `__inline__ static __device__ size_t4 make_index (size_t v0)`
- `__inline__ static __device__ size_t4 make_index (size_t2 v0)`
- `__inline__ static __device__ size_t4 make_index (size_t3 v0)`
- `__inline__ static __device__ size_t4 make_index (size_t4 v0)`
- `template<typename T2 > __device__ static __forceinline__ void * create (type< T2 >, void *v)`
- `template<typename T2, int Dim2> __device__ static __forceinline__ void * create (type< bufferId< T2, Dim2 >, void *v)`

2.6 optix::prime::BufferDescObj Class Reference

2.6.1 Detailed Description

Encapsulates an OptiX Prime buffer descriptor. The purpose of a buffer descriptor is to provide information about a buffer's type, format, and location. It also describes the region of the buffer to use. Inheritance diagram for optix::prime::BufferDescObj:



Public Member Functions

- [Context](#) `getContext ()`
- void `setRange` (RTPsize begin, RTPsize end)
- void `setStride` (unsigned strideBytes)
- void `setCudaDeviceNumber` (unsigned deviceNumber)
- [RTPbufferdesc](#) `getRTPbufferdesc ()`

Friends

- class **ContextObj**
- class **ModelObj**
- class **QueryObj**

2.7 optix::bufferId< T, Dim > Struct Template Reference

2.7.1 Detailed Description

`template<typename T, int Dim>struct optix::bufferId< T, Dim >`

`bufferId` is a host version of the device side `bufferId`.

Use `bufferId` to define types that can be included from both the host and device code. This class provides a container that can be used to transport the buffer id back and forth between host and device code. The `bufferId` class is useful, because it can take a buffer id obtained from `rtBufferGetId` and provide accessors similar to the `buffer` class.

"bindless_type.h" used by both host and device code:

```
#include <optix_world.h>
struct BufInfo {
    int val;
    rtBufferId<int, 1> data;
};
```

Host code:

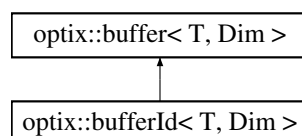
```
#include "bindless_type.h"
BufInfo input_buffer_info;
input_buffer_info.val = 0;
input_buffer_info.data = rtBufferId<int,1>(inputBuffer0->getId());
context["input_buffer_info"]->setUserData(sizeof(BufInfo), &input_buffer_info);
```

Device code:

```
#include "bindless_type.h"
rtBuffer<int,1> result;
rtDeclareVariable(BufInfo, input_buffer_info, ,);

RT_PROGRAM void bindless()
{
    int value = input_buffer_info.data[input_buffer_info.val];
    result[0] = value;
}
```

Inheritance diagram for `optix::bufferId< T, Dim >`:



Public Types

- typedef `buffer`< T, Dim >
::`WrapperType` **WrapperType**
- typedef `buffer`< T, Dim >::`IndexType` **IndexType**

Public Member Functions

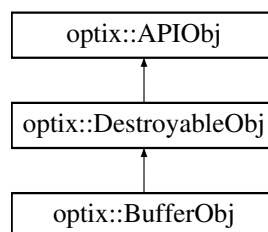
- `__device__ __forceinline__` **bufferId** (`RTbufferidnull` nullid)
- `__device__ __forceinline__` **bufferId** (int id)
- `__device__ __forceinline__`
bufferId & **operator=** (`RTbufferidnull` nullid)
- `__device__ __forceinline__`
`IndexType` **size** () const
- `__device__ __forceinline__` T & **operator[]** (`IndexType` i) const
- `__device__ __forceinline__` int **getId** () const
- `__device__ __forceinline__` **operator bool** () const
- **bufferId** (int id)
- int **getId** () const

Additional Inherited Members

2.8 optix::BufferObj Class Reference

2.8.1 Detailed Description

Buffer wraps the OptiX C API `RTbuffer` opaque type and its associated function set. Inheritance diagram for `optix::BufferObj`:



Public Member Functions

- void **destroy** ()
- void **validate** ()
- `Context` **getContext** () const
- `RTbuffer` **get** ()
- void **setFormat** (`RTformat` format)
- `RTformat` **getFormat** () const
- void **setElementSize** (`RTsize` size_of_element)
- `RTsize` **getElementSize** () const
- void **getDevicePointer** (int optix_device_ordinal, void **device_pointer)
- void * **getDevicePointer** (int optix_device_ordinal)
- void **setDevicePointer** (int optix_device_ordinal, void *device_pointer)
- void **markDirty** ()
- void **setSize** (`RTsize` width)

- void [getSize](#) (RTsize &width) const
- void [getMipLevelSize](#) (unsigned int level, RTsize &width) const
- void [setSize](#) (RTsize width, RTsize height)
- void [getSize](#) (RTsize &width, RTsize &height) const
- void [getMipLevelSize](#) (unsigned int level, RTsize &width, RTsize &height) const
- void [setSize](#) (RTsize width, RTsize height, RTsize depth)
- void [getSize](#) (RTsize &width, RTsize &height, RTsize &depth) const
- void [getMipLevelSize](#) (unsigned int level, RTsize &width, RTsize &height, RTsize &depth) const
- void [setSize](#) (unsigned int dimensionality, const RTsize *dims)
- void [getSize](#) (unsigned int dimensionality, RTsize *dims) const
- unsigned int [getDimensionality](#) () const
- void [setMipLevelCount](#) (unsigned int levels)
- unsigned int [getMipLevelCount](#) () const

- int [getId](#) () const

- unsigned int [getGLBOId](#) () const
- void [registerGLBuffer](#) ()
- void [unregisterGLBuffer](#) ()

- void [setAttribute](#) (RTbufferattribute attrib, RTsize size, void *p)
- void [getAttribute](#) (RTbufferattribute attrib, RTsize size, void *p)

- void * [map](#) (unsigned int level=0, unsigned int map_flags=RT_BUFFER_MAP_READ_WRITE, void *user_ - owned=0)
- void [unmap](#) (unsigned int level=0)

- void [bindProgressiveStream](#) (Buffer source)
- void [getProgressiveUpdateReady](#) (int *ready, unsigned int *subframe_count, unsigned int *max_subframes)
- bool [getProgressiveUpdateReady](#) ()
- bool [getProgressiveUpdateReady](#) (unsigned int &subframe_count)
- bool [getProgressiveUpdateReady](#) (unsigned int &subframe_count, unsigned int &max_subframes)

Friends

- class **Handle**< **BufferObj** >

Additional Inherited Members

2.8.2 Member Function Documentation

2.8.2.1 void optix::BufferObj::bindProgressiveStream (Buffer source) [inline]

Bind a buffer as source for a progressive stream. See [rtBufferBindProgressiveStream](#).

2.8.2.2 void * optix::BufferObj::getDevicePointer (int optix_device_ordinal) [inline]

Set the data format for the buffer. See [rtBufferSetFormat](#).

2.8.2.3 unsigned int optix::BufferObj::getGLBOId () const [inline]

Queries the OpenGL Buffer Object ID associated with this buffer. See [rtBufferGetGLBOId](#).

2.8.2.4 int optix::BufferObj::getId () const [inline]

Queries an id suitable for referencing the buffer in an another buffer. See [rtBufferGetId](#).

2.8.2.5 void * optix::BufferObj::map (unsigned int *level* = 0, unsigned int *map_flags* = RT_BUFFER_MAP_READ_WRITE, void * *user_owned* = 0) [inline]

Maps a buffer object for host access. See [rtBufferMap](#) and [rtBufferMapEx](#).

2.8.2.6 void optix::BufferObj::setAttribute (RTbufferattribute *attrib*, RTsize *size*, void * *p*) [inline]

Set a Buffer Attribute. See [rtBufferSetAttribute](#).

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

Set the data format for the buffer. See [rtBufferSetFormat](#).

2.8.2.8 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](#).

2.9 optix::callableProgramId< T > Class Template Reference

2.10 rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T > Class Template Reference

Public Member Functions

- `__device__ __forceinline__ ReturnT operator() ()`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1, Arg2T arg2)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4, Arg5T arg5)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4, Arg5T arg5, Arg6T arg6)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4, Arg5T arg5, Arg6T arg6, Arg7T arg7)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4, Arg5T arg5, Arg6T arg6, Arg7T arg7, Arg8T arg8)`
- `__device__ __forceinline__ ReturnT operator() (Arg0T arg0, Arg1T arg1, Arg2T arg2, Arg3T arg3, Arg4T arg4, Arg5T arg5, Arg6T arg6, Arg7T arg7, Arg8T arg8, Arg9T arg9)`

Protected Attributes

- `int m_id`

2.11 rti_internal_callableprogram::check_is_CPArgVoid< Condition, Dummy > Struct Template Reference

Public Types

- `typedef bool result`

2.12 rti_internal_callableprogram::check_is_CPArgVoid< false, IntentionalError > Struct Template Reference

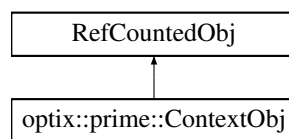
Public Types

- typedef
IntentionalError::does_not_exist **result**

2.13 optix::prime::ContextObj Class Reference

2.13.1 Detailed Description

Wraps the OptiX Prime C API [RTPcontext](#) opaque type and its associated function set representing an OptiX Prime context. Inheritance diagram for `optix::prime::ContextObj`:



Public Member Functions

- [BufferDesc](#) `createBufferDesc` ([RTPbufferformat](#) format, [RTPbuffertype](#) type, void *buffer)
- [Model](#) `createModel` ()
- void [setCudaDeviceNumbers](#) (const std::vector< unsigned > &deviceNumbers)
- void [setCudaDeviceNumbers](#) (unsigned deviceCount, const unsigned *deviceNumbers)
- void [setCpuThreads](#) (unsigned numThreads)
- std::string [getLastErrorString](#) ()
- [RTPcontext](#) `getRTPcontext` ()

Static Public Member Functions

- static [Context](#) `create` ([RTPcontexttype](#) type)

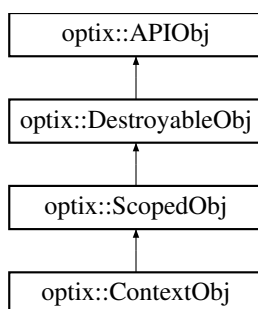
Friends

- class **QueryObj**
- class **ModelObj**
- class **BufferDescObj**

2.14 optix::ContextObj Class Reference

2.14.1 Detailed Description

Context object wraps the OptiX C API `RTcontext` opaque type and its associated function set. Inheritance diagram for `optix::ContextObj`:



Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context](#) [getContext](#) () const
- void [compile](#) ()
- void [setRemoteDevice](#) ([RemoteDevice](#) remote_device)
- int [getRunningState](#) () const
- [RTcontext](#) [get](#) ()

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

- [Acceleration](#) [createAcceleration](#) (const std::string &builder, const std::string &ignored="")
- [Buffer](#) [createBuffer](#) (unsigned int type)
- [Buffer](#) [createBuffer](#) (unsigned int type, [RTformat](#) format)
- [Buffer](#) [createBuffer](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width)
- [Buffer](#) [createMipmappedBuffer](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width, unsigned int levels)
- [Buffer](#) [createBuffer](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width, [RTsize](#) height)
- [Buffer](#) [createMipmappedBuffer](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width, [RTsize](#) height, unsigned int levels)
- [Buffer](#) [createBuffer](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width, [RTsize](#) height, [RTsize](#) depth)
- [Buffer](#) [createMipmappedBuffer](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width, [RTsize](#) height, [RTsize](#) depth, unsigned int levels)
- [Buffer](#) [create1DLayeredBuffer](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width, [RTsize](#) layers, unsigned int levels)
- [Buffer](#) [create2DLayeredBuffer](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width, [RTsize](#) height, [RTsize](#) layers, unsigned int levels)
- [Buffer](#) [createCubeBuffer](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width, [RTsize](#) height, unsigned int levels)
- [Buffer](#) [createCubeLayeredBuffer](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width, [RTsize](#) height, [RTsize](#) faces, unsigned int levels)
- [Buffer](#) [createBufferForCUDA](#) (unsigned int type)
- [Buffer](#) [createBufferForCUDA](#) (unsigned int type, [RTformat](#) format)
- [Buffer](#) [createBufferForCUDA](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width)
- [Buffer](#) [createBufferForCUDA](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width, [RTsize](#) height)
- [Buffer](#) [createBufferForCUDA](#) (unsigned int type, [RTformat](#) format, [RTsize](#) width, [RTsize](#) height, [RTsize](#) depth)
- [Buffer](#) [createBufferFromGLBO](#) (unsigned int type, unsigned int vbo)
- [TextureSampler](#) [createTextureSamplerFromGLImage](#) (unsigned int id, [RTgltarget](#) target)
- [Buffer](#) [getBufferFromId](#) (int buffer_id)
- [Program](#) [getProgramFromId](#) (int program_id)
- [TextureSampler](#) [getTextureSamplerFromId](#) (int sampler_id)
- [Geometry](#) [createGeometry](#) ()

- [GeometryInstance createGeometryInstance](#) ()
- `template<class Iterator >`
[GeometryInstance createGeometryInstance](#) ([Geometry](#) geometry, Iterator matlbegin, Iterator matlend)
- [Group createGroup](#) ()
- `template<class Iterator >`
[Group createGroup](#) (Iterator childbegin, Iterator childend)
- [GeometryGroup createGeometryGroup](#) ()
- `template<class Iterator >`
[GeometryGroup createGeometryGroup](#) (Iterator childbegin, Iterator childend)
- [Transform createTransform](#) ()
- [Material createMaterial](#) ()
- [Program createProgramFromPTXFile](#) (const std::string &ptx, const std::string &program_name)
- [Program createProgramFromPTXString](#) (const std::string &ptx, const std::string &program_name)
- [Selector createSelector](#) ()
- [TextureSampler createTextureSampler](#) ()

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

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

- void [setCPUNumThreads](#) (int cpu_num_threads)
- void [setGPUPagingForcedOff](#) (int gpu_paging_forced_off)
- `template<class T >`
void [setAttribute](#) (RTcontextattribute attribute, const T &val)

- void [setStackSize](#) (RTsize stack_size_bytes)
- RTsize [getStackSize](#) () const
- void [setTimeoutCallback](#) (RTtimeoutcallback callback, double min_polling_seconds)
- void [setEntryPointCount](#) (unsigned int num_entry_points)
- unsigned int [getEntryPointCount](#) () const
- void [setRayTypeCount](#) (unsigned int num_ray_types)
- unsigned int [getRayTypeCount](#) () const

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

- void [launch](#) (unsigned int entry_point_index, RTsize image_width)
- void [launch](#) (unsigned int entry_point_index, RTsize image_width, RTsize image_height)
- void [launch](#) (unsigned int entry_point_index, RTsize image_width, RTsize image_height, RTsize image_depth)

- void [launchProgressive](#) (unsigned int entry_point_index, RTsize image_width, RTsize image_height, unsigned int max_subframes)
- void [stopProgressive](#) ()
- void [setPrintEnabled](#) (bool enabled)
- bool [getPrintEnabled](#) () const
- void [setPrintBufferSize](#) (RTsize buffer_size_bytes)
- RTsize [getPrintBufferSize](#) () const
- void [setPrintLaunchIndex](#) (int x, int y=-1, int z=-1)
- `optix::int3` [getPrintLaunchIndex](#) () const
- [Variable](#) [declareVariable](#) (const std::string &name)
- [Variable](#) [queryVariable](#) (const std::string &name) const
- void [removeVariable](#) ([Variable](#) v)
- unsigned int [getVariableCount](#) () const
- [Variable](#) [getVariable](#) (unsigned int index) const

Static Public Member Functions

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

Friends

- class **Handle**< **ContextObj** >

2.14.2 Member Function Documentation

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

See [APIObj::checkError](#)

Reimplemented from [optix::APIObj](#).

2.14.2.2 **Buffer** `optix::ContextObj::create1DLayeredBuffer (unsigned int type, RTformat format, RTsize width, RTsize layers, unsigned int levels)` `[inline]`

Create a 1D layered mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#), [rtBufferSetMipLevelCount](#), and [rtBufferSetSize3D](#).

2.14.2.3 **Buffer** `optix::ContextObj::create2DLayeredBuffer (unsigned int type, RTformat format, RTsize width, RTsize height, RTsize layers, unsigned int levels)` `[inline]`

Create a 2D layered mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#), [rtBufferSetMipLevelCount](#), and [rtBufferSetSize3D](#).

2.14.2.4 **Acceleration** `optix::ContextObj::createAcceleration (const std::string & builder, const std::string & ignored = " ")` `[inline]`

traverser parameter unused in OptiX 4.0 See [rtAccelerationCreate](#).

2.14.2.5 **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](#).

2.14.2.6 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](#).

2.14.2.7 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](#).

2.14.2.8 Buffer `optix::ContextObj::createBufferForCUDA (unsigned int type, RTformat format, RTsize width)`
`[inline]`

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize1D](#).

2.14.2.9 Buffer `optix::ContextObj::createBufferForCUDA (unsigned int type, RTformat format, RTsize width, RTsize height)` `[inline]`

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize2D](#).

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

Create a buffer for CUDA with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize3D](#).

2.14.2.11 Buffer `optix::ContextObj::createCubeBuffer (unsigned int type, RTformat format, RTsize width, RTsize height, unsigned int levels)` `[inline]`

Create a cube mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#), [rtBufferSetMipLevelCount](#), and [rtBufferSetSize3D](#).

2.14.2.12 Buffer `optix::ContextObj::createCubeLayeredBuffer (unsigned int type, RTformat format, RTsize width, RTsize height, RTsize faces, unsigned int levels)` `[inline]`

Create a cube layered mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#), [rtBufferSetMipLevelCount](#), and [rtBufferSetSize3D](#).

2.14.2.13 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](#)

2.14.2.14 template<class Iterator > GeometryInstance `optix::ContextObj::createGeometryInstance (Geometry geometry, Iterator matlbegin, Iterator matlend)`

Create a geometry instance with a Geometry object and a set of associated materials. See [rtGeometryInstanceCreate](#), [rtGeometryInstanceSetMaterialCount](#), and [rtGeometryInstanceSetMaterial](#)

2.14.2.15 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](#)

2.14.2.16 Buffer `optix::ContextObj::createMipmappedBuffer (unsigned int type, RTformat format, RTsize width, unsigned int levels) [inline]`

Create a mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize1DMipmapped](#).

2.14.2.17 Buffer `optix::ContextObj::createMipmappedBuffer (unsigned int type, RTformat format, RTsize width, RTsize height, unsigned int levels) [inline]`

Create a mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize2DMipmapped](#).

2.14.2.18 Buffer `optix::ContextObj::createMipmappedBuffer (unsigned int type, RTformat format, RTsize width, RTsize height, RTsize depth, unsigned int levels) [inline]`

Create a mipmapped buffer with given RTbuffertype, RTformat and dimension. See [rtBufferCreate](#), [rtBufferSetFormat](#) and [rtBufferSetSize3DMipmapped](#).

2.14.2.19 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](#).

2.14.2.20 Buffer `optix::ContextObj::getBufferFromId (int buffer_id) [inline]`

Queries the Buffer object from a given buffer id obtained from a previous call to [BufferObj::getId](#). See [BufferObj::getId](#) and [rtContextGetBufferFromId](#).

2.14.2.21 Context `optix::ContextObj::getContext () const [inline],[virtual]`

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

Implements [optix::APIObj](#).

2.14.2.22 `unsigned int optix::ContextObj::getEnabledDeviceCount () const [inline]`

See [rtContextGetDeviceCount](#). As opposed to `getDeviceCount`, this returns only the number of enabled devices.

2.14.2.23 `int optix::ContextObj::getMaxTextureCount () const [inline]`

See [rtContextGetAttribute](#)

2.14.2.24 Program `optix::ContextObj::getProgramFromId (int program_id) [inline]`

Queries the Program object from a given program id obtained from a previous call to [ProgramObj::getId](#). See [ProgramObj::getId](#) and [rtContextGetProgramFromId](#).

2.14.2.25 TextureSampler `optix::ContextObj::getTextureSamplerFromId (int sampler_id) [inline]`

Queries the TextureSampler object from a given sampler id obtained from a previous call to [TextureSamplerObj::getId](#). See [TextureSamplerObj::getId](#) and [rtContextGetTextureSamplerFromId](#).

2.14.2.26 `unsigned int optix::ContextObj::getVariableCount () const [inline],[virtual]`

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

Implements [optix::ScopedObj](#).

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

See [rtContextLaunch](#)

2.14.2.28 void optix::ContextObj::launchProgressive (unsigned int *entry_point_index*, RTsize *image_width*, RTsize *image_height*, unsigned int *max_subframes*) [inline]

See [rtContextLaunchProgressive](#)

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

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

Implements [optix::ScopedObj](#).

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

See [rtContextSetAttribute](#)

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

See [rtContextSetDevices](#)

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

See [rtContextSetPrintEnabled](#)

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

See [rtContextSetRayGenerationProgram](#)

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

See [rtContextSetStackSize](#)

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

See [rtContextSetTimeoutCallback](#) RTtimeoutcallback is defined as typedef int (*RTtimeoutcallback)(void).

2.15 rti_internal_callableprogram::CPArgVoid Class Reference

2.16 optix::DestroyableObj Class Reference

2.16.1 Detailed Description

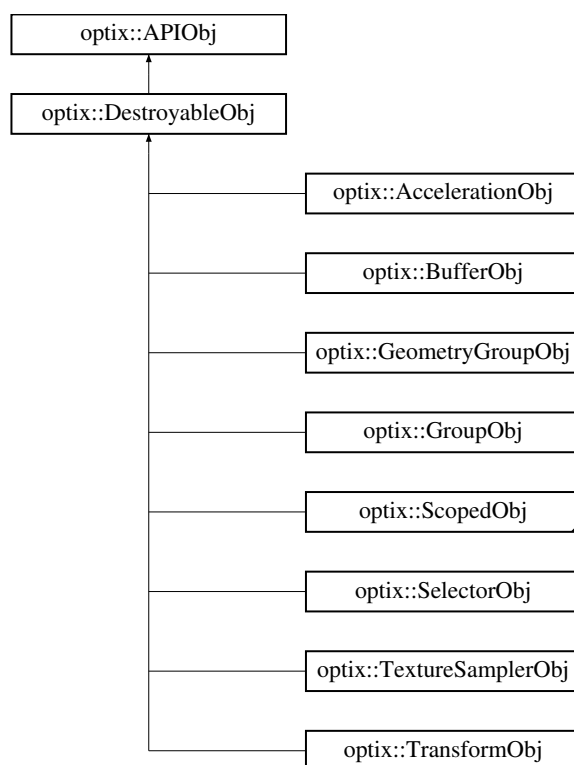
Base class for all wrapper objects which can be destroyed and validated.

Wraps:

- RTcontext
- RTgeometry
- RTgeometryinstance
- RTgeometrygroup
- RTgroup

- RTmaterial
- RTprogram
- RTselector
- RTtexturesampler
- RTtransform

Inheritance diagram for optix::DestroyableObj:



Public Member Functions

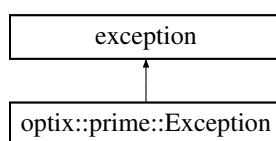
- virtual void `destroy` ()=0
- virtual void `validate` ()=0

Additional Inherited Members

2.17 optix::prime::Exception Class Reference

2.17.1 Detailed Description

Encapsulates an OptiX Prime exception. Inheritance diagram for optix::prime::Exception:



Public Member Functions

- [RTPresult getErrorCode](#) () const
- const std::string & [getErrorString](#) () const
- virtual const char * [what](#) () const throw ()

Static Public Member Functions

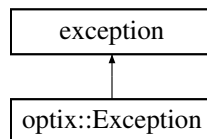
- static [Exception makeException](#) (RTPresult code)
- static [Exception makeException](#) (RTPresult code, RTPcontext context)

2.18 optix::Exception Class Reference

2.18.1 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 rt-ContextGetErrorString call. Inheritance diagram for optix::Exception:



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)

2.18.2 Constructor & Destructor Documentation

2.18.2.1 virtual optix::Exception::~~Exception () throw () [inline],[virtual]

Virtual destructor (needed for virtual function calls inherited from std::exception).

2.18.3 Member Function Documentation

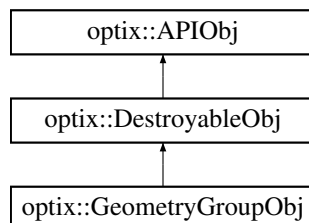
2.18.3.1 [Exception](#) optix::Exception::makeException (RTresult code, RTcontext context) [inline],[static]

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

2.19 optix::GeometryGroupObj Class Reference

2.19.1 Detailed Description

GeometryGroup wraps the OptiX C API RTgeometrygroup opaque type and its associated function set. Inheritance diagram for optix::GeometryGroupObj:



Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context](#) [getContext](#) () const
- [RTgeometrygroup](#) [get](#) ()

- void [setAcceleration](#) ([Acceleration](#) acceleration)
- [Acceleration](#) [getAcceleration](#) () const

- void [setChildCount](#) (unsigned int count)
- unsigned int [getChildCount](#) () const
- void [setChild](#) (unsigned int index, [GeometryInstance](#) geometryinstance)
- [GeometryInstance](#) [getChild](#) (unsigned int index) const
- unsigned int [addChild](#) ([GeometryInstance](#) child)
- unsigned int [removeChild](#) ([GeometryInstance](#) child)
- void [removeChild](#) (int index)
- void [removeChild](#) (unsigned int index)
- unsigned int [getChildIndex](#) ([GeometryInstance](#) child) const

Friends

- class **Handle**< **GeometryGroupObj** >

Additional Inherited Members

2.19.2 Member Function Documentation

2.19.2.1 unsigned int optix::GeometryGroupObj::removeChild ([GeometryInstance](#) *child*) [\[inline\]](#)

Remove a child in this group. Note: this function is not order-preserving. Returns the position of the removed element if succeeded. Throws [RT_ERROR_INVALID_VALUE](#) if the parameter is invalid.

2.19.2.2 void optix::GeometryGroupObj::removeChild (int *index*) [\[inline\]](#)

Remove a child in this group. Note: this function is not order-preserving. Throws [RT_ERROR_INVALID_VALUE](#) if the parameter is invalid.

2.19.2.3 void optix::GeometryGroupObj::removeChild (unsigned int *index*) [inline]

Remove a child in this group. Note: this function is not order-preserving. Throws [RT_ERROR_INVALID_VALUE](#) if the parameter is invalid.

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

Set the Acceleration structure for this group. See [rtGeometryGroupSetAcceleration](#).

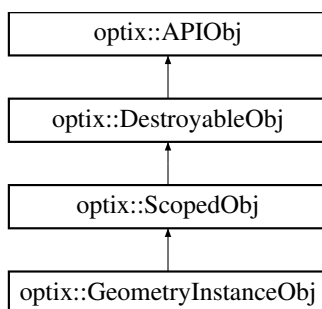
2.19.2.5 void optix::GeometryGroupObj::setChildCount (unsigned int *count*) [inline]

Set the number of children for this group. See [rtGeometryGroupSetChildCount](#).

2.20 optix::GeometryInstanceObj Class Reference

2.20.1 Detailed Description

GeometryInstance wraps the OptiX C API RTgeometryinstance acceleration opaque type and its associated function set. Inheritance diagram for optix::GeometryInstanceObj:



Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context](#) [getContext](#) () const
- [RTgeometryinstance](#) [get](#) ()
- void [setGeometry](#) ([Geometry](#) geometry)
- [Geometry](#) [getGeometry](#) () const
- void [setMaterialCount](#) (unsigned int count)
- unsigned int [getMaterialCount](#) () const
- void [setMaterial](#) (unsigned int idx, [Material](#) material)
- [Material](#) [getMaterial](#) (unsigned int idx) const
- unsigned int [addMaterial](#) ([Material](#) material)
- [Variable](#) [declareVariable](#) (const std::string &name)
- [Variable](#) [queryVariable](#) (const std::string &name) const
- void [removeVariable](#) ([Variable](#) v)
- unsigned int [getVariableCount](#) () const
- [Variable](#) [getVariable](#) (unsigned int index) const

Friends

- class [Handle](#)< [GeometryInstanceObj](#) >

Additional Inherited Members

2.20.2 Member Function Documentation

2.20.2.1 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`.

2.20.2.2 `unsigned int optix::GeometryInstanceObj::getVariableCount () const` `[inline]`, `[virtual]`

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

Implements `optix::ScopedObj`.

2.20.2.3 Variable `optix::GeometryInstanceObj::queryVariable (const std::string & name) const` `[inline]`, `[virtual]`

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

Implements `optix::ScopedObj`.

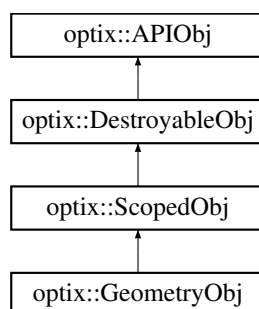
2.20.2.4 `void optix::GeometryInstanceObj::setGeometry (Geometry geometry)` `[inline]`

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

2.21 optix::GeometryObj Class Reference

2.21.1 Detailed Description

Geometry wraps the OptiX C API `RTgeometry` opaque type and its associated function set. Inheritance diagram for `optix::GeometryObj`:



Public Member Functions

- void `destroy` ()
- void `validate` ()
- `Context` `getContext` () const
- `RTgeometry` `get` ()

- void `markDirty` ()
- bool `isDirty` () const

- void [setPrimitiveCount](#) (unsigned int num_primitives)
- unsigned int [getPrimitiveCount](#) () const
- void [setPrimitiveIndexOffset](#) (unsigned int index_offset)
- unsigned int [getPrimitiveIndexOffset](#) () const
- void [setBoundingBoxProgram](#) ([Program](#) program)
- [Program](#) [getBoundingBoxProgram](#) () const
- void [setIntersectionProgram](#) ([Program](#) program)
- [Program](#) [getIntersectionProgram](#) () const
- [Variable](#) [declareVariable](#) (const std::string &name)
- [Variable](#) [queryVariable](#) (const std::string &name) const
- void [removeVariable](#) ([Variable](#) v)
- unsigned int [getVariableCount](#) () const
- [Variable](#) [getVariable](#) (unsigned int index) const

Friends

- class **Handle**< **GeometryObj** >

Additional Inherited Members

2.21.2 Member Function Documentation

2.21.2.1 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](#).

2.21.2.2 unsigned int `optix::GeometryObj::getPrimitiveCount () const [inline]`

Query the number of primitives in this geometry object (eg, number of triangles in mesh). See [rtGeometryGetPrimitiveCount](#)

2.21.2.3 unsigned int `optix::GeometryObj::getPrimitiveIndexOffset () const [inline]`

Query the primitive index offset for this geometry object. See [rtGeometryGetPrimitiveIndexOffset](#)

2.21.2.4 unsigned int `optix::GeometryObj::getVariableCount () const [inline],[virtual]`

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

Implements [optix::ScopedObj](#).

2.21.2.5 void `optix::GeometryObj::markDirty () [inline]`

Deprecated in OptiX 4.0 See [rtGeometryMarkDirty](#).

2.21.2.6 Variable `optix::GeometryObj::queryVariable (const std::string & name) const [inline],[virtual]`

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

Implements [optix::ScopedObj](#).

2.21.2.7 void optix::GeometryObj::setBoundingBoxProgram (Program *program*) [inline]

Set the bounding box program for this geometry. See [rtGeometrySetBoundingBoxProgram](#).

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

Set the number of primitives in this geometry object (eg, number of triangles in mesh). See [rtGeometrySetPrimitiveCount](#)

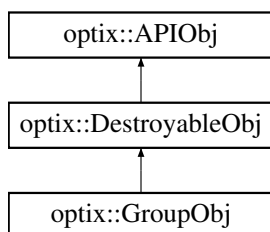
2.21.2.9 void optix::GeometryObj::setPrimitiveIndexOffset (unsigned int *index_offset*) [inline]

Set the primitive index offset for this geometry object. See [rtGeometrySetPrimitiveIndexOffset](#)

2.22 optix::GroupObj Class Reference

2.22.1 Detailed Description

Group wraps the OptiX C API RTgroup opaque type and its associated function set. Inheritance diagram for optix::GroupObj:



Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- Context [getContext](#) () const
- RTgroup [get](#) ()

- void [setAcceleration](#) (Acceleration acceleration)
- Acceleration [getAcceleration](#) () const

- void [setChildCount](#) (unsigned int count)
- unsigned int [getChildCount](#) () const
- template<typename T >
void [setChild](#) (unsigned int index, T child)
- template<typename T >
T [getChild](#) (unsigned int index) const
- RObjecttype [getChildType](#) (unsigned int index) const
- template<typename T >
unsigned int [addChild](#) (T child)
- template<typename T >
unsigned int [removeChild](#) (T child)
- void [removeChild](#) (int index)
- void [removeChild](#) (unsigned int index)
- template<typename T >
unsigned int [getChildIndex](#) (T child) const

Friends

- class `Handle< GroupObj >`

Additional Inherited Members

2.22.2 Member Function Documentation

2.22.2.1 `template<typename T> unsigned int optix::GroupObj::removeChild (T child) [inline]`

Remove a child in this group. Note: this function is not order-preserving. Returns the position of the removed element if succeeded. Throws `RT_ERROR_INVALID_VALUE` if the parameter is invalid.

2.22.2.2 `void optix::GroupObj::removeChild (int index) [inline]`

Remove a child in this group. Note: this function is not order-preserving. Throws `RT_ERROR_INVALID_VALUE` if the parameter is invalid.

2.22.2.3 `void optix::GroupObj::removeChild (unsigned int index) [inline]`

Remove a child in this group. Note: this function is not order-preserving. Throws `RT_ERROR_INVALID_VALUE` if the parameter is invalid.

2.22.2.4 `void optix::GroupObj::setAcceleration (Acceleration acceleration) [inline]`

Set the Acceleration structure for this group. See `rtGroupSetAcceleration`.

2.22.2.5 `void optix::GroupObj::setChildCount (unsigned int count) [inline]`

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

2.23 `optix::Handle< T >` Class Template Reference

2.23.1 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.

Public Member Functions

- `Handle ()`
- `Handle (T *ptr)`
- `template<class U> Handle (U *ptr)`
- `Handle (const Handle< T > ©)`
- `template<class U> Handle (const Handle< U > ©)`
- `Handle< T > & operator= (const Handle< T > ©)`
- `template<class U> Handle< T > & operator= (const Handle< U > ©)`
- `~Handle ()`
- `T * operator-> ()`
- `const T * operator-> () const`

- `T * get ()`
- `const T * get () const`
- `operator bool () const`
- `Handle< VariableObj > operator\[\] (const std::string &varname)`
- `Handle< VariableObj > operator\[\] (const char *varname)`

Static Public Member Functions

- static `Handle< T > take (typename T::api_t p)`
- static `Handle< T > take (RObject p)`
- static `Handle< T > create ()`
- static `Handle< T > create (const std::string &a, const std::string &b, const std::string &c)`
- static unsigned int `getDeviceCount ()`

2.23.2 Member Function Documentation

2.23.2.1 `template<class T> Handle< VariableObj > optix::Handle< T >::operator[] (const std::string & varname)`

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

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

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

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

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

can be interpreted as either

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

or

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

2.23.2.3 `template<class T> static Handle<T> optix::Handle< T >::take (RObject p) [inline],[static]`

Special version that takes an RObject which must be cast up to the appropriate OptiX API opaque type.

2.24 rti_internal_callableprogram::is_CPArgVoid< T1 > Struct Template Reference

Static Public Attributes

- static const bool **result** = false

2.25 rti_internal_callableprogram::is_CPArgVoid< CPArgVoid > Struct Template Reference

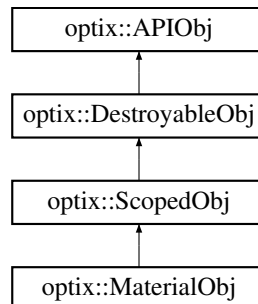
Static Public Attributes

- static const bool **result** = true

2.26 optix::MaterialObj Class Reference

2.26.1 Detailed Description

Material wraps the OptiX C API RTmaterial opaque type and its associated function set. Inheritance diagram for optix::MaterialObj:



Public Member Functions

- void [destroy](#) ()
- void [validate](#) ()
- [Context](#) [getContext](#) () const
- [RTmaterial](#) [get](#) ()
- void [setClosestHitProgram](#) (unsigned int ray_type_index, [Program](#) program)
- [Program](#) [getClosestHitProgram](#) (unsigned int ray_type_index) const
- void [setAnyHitProgram](#) (unsigned int ray_type_index, [Program](#) program)
- [Program](#) [getAnyHitProgram](#) (unsigned int ray_type_index) const
- [Variable](#) [declareVariable](#) (const std::string &name)
- [Variable](#) [queryVariable](#) (const std::string &name) const
- void [removeVariable](#) ([Variable](#) v)
- unsigned int [getVariableCount](#) () const
- [Variable](#) [getVariable](#) (unsigned int index) const

Friends

- class **Handle**< **MaterialObj** >

Additional Inherited Members

2.26.2 Member Function Documentation

2.26.2.1 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](#).

2.26.2.2 unsigned int optix::MaterialObj::getVariableCount () const [inline], [virtual]

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

Implements [optix::ScopedObj](#).

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

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

Implements `optix::ScopedObj`.

2.26.2.4 `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`.

2.27 `optix::Matrix< M, N >` Class Template Reference

2.27.1 Detailed Description

`template<unsigned int M, unsigned int N>class optix::Matrix< M, N >`

A matrix with M rows and N columns.

Description

`Matrix` provides a utility class for small-dimension floating-point matrices, such as transformation matrices. `Matrix` may also be useful in other computation and can be used in both host and device code. Typedefs are provided for 2x2 through 4x4 matrices.

History

`Matrix` was introduced in OptiX 1.0.

See also `rtVariableSetMatrix*`

Public Types

- typedef `VectorDim< N >::VectorType floatN`
- typedef `VectorDim< M >::VectorType floatM`

Public Member Functions

- `RT_HOSTDEVICE Matrix ()`
- `RT_HOSTDEVICE Matrix (const float data[M *N])`
- `RT_HOSTDEVICE Matrix (const Matrix &m)`
- `RT_HOSTDEVICE Matrix & operator= (const Matrix &b)`
- `RT_HOSTDEVICE float operator[] (unsigned int i) const`
- `RT_HOSTDEVICE float & operator[] (unsigned int i)`
- `RT_HOSTDEVICE floatN getRow (unsigned int m) const`
- `RT_HOSTDEVICE floatM getCol (unsigned int n) const`
- `RT_HOSTDEVICE float * getData ()`
- `RT_HOSTDEVICE const float * getData () const`
- `RT_HOSTDEVICE void setRow (unsigned int m, const floatN &r)`
- `RT_HOSTDEVICE void setCol (unsigned int n, const floatM &c)`
- `RT_HOSTDEVICE Matrix< N, M > transpose () const`
- `RT_HOSTDEVICE Matrix< 4, 4 > inverse () const`
- `RT_HOSTDEVICE float det () const`
- `RT_HOSTDEVICE bool operator< (const Matrix< M, N > &rhs) const`
- `template<> OPTIXU_INLINE RT_HOSTDEVICE float det () const`
- `template<> OPTIXU_INLINE RT_HOSTDEVICE float det () const`

- template<>
OPTIXU_INLINE RT_HOSTDEVICE
[Matrix](#)< 4, 4 > **inverse** () const
- template<>
OPTIXU_INLINE RT_HOSTDEVICE
[Matrix](#)< 4, 4 > **rotate** (const float radians, const float3 &axis)
- template<>
OPTIXU_INLINE RT_HOSTDEVICE
[Matrix](#)< 4, 4 > **translate** (const float3 &vec)
- template<>
OPTIXU_INLINE RT_HOSTDEVICE
[Matrix](#)< 4, 4 > **scale** (const float3 &vec)
- template<>
OPTIXU_INLINE RT_HOSTDEVICE
[Matrix](#)< 4, 4 > **fromBasis** (const float3 &u, const float3 &v, const float3 &w, const float3 &c)

Static Public Member Functions

- static RT_HOSTDEVICE [Matrix](#)< 4, 4 > **rotate** (const float radians, const float3 &axis)
- static RT_HOSTDEVICE [Matrix](#)< 4, 4 > **translate** (const float3 &vec)
- static RT_HOSTDEVICE [Matrix](#)< 4, 4 > **scale** (const float3 &vec)
- static RT_HOSTDEVICE [Matrix](#)< 4, 4 > **fromBasis** (const float3 &u, const float3 &v, const float3 &w, const float3 &c)
- static RT_HOSTDEVICE [Matrix](#)< N, N > **identity** ()

2.27.2 Constructor & Destructor Documentation

2.27.2.1 template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE optix::Matrix< M, N >::Matrix ()

A column of the matrix.

Create an uninitialized matrix

2.27.2.2 template<unsigned int M, unsigned int N> RT_HOSTDEVICE optix::Matrix< M, N >::Matrix (const float *data*[M *N]) [inline],[explicit]

Create a matrix from the specified float array

2.27.2.3 template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE optix::Matrix< M, N >::Matrix (const Matrix< M, N > & *m*)

Copy the matrix

2.27.3 Member Function Documentation

2.27.3.1 template<unsigned int M, unsigned int N> RT_HOSTDEVICE float optix::Matrix< M, N >::det () const

Returns the determinant of the matrix

2.27.3.2 template<unsigned int M, unsigned int N> static RT_HOSTDEVICE Matrix<4,4> optix::Matrix< M, N >::fromBasis (const float3 & *u*, const float3 & *v*, const float3 & *w*, const float3 & *c*) [static]

Creates a matrix from an ONB and center point

2.27.3.3 template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE Matrix< M, N >::floatM optix::Matrix< M, N >::getCol (unsigned int *n*) const

Access the specified column 0..N. Returns float, float2, float3 or float4 depending on the matrix size

2.27.3.4 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE float * optix::Matrix< M, N >::getData ()`

Returns a pointer to the internal data array. The data array is stored in row-major order.

2.27.3.5 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE const float * optix::Matrix< M, N >::getData () const`

Returns a const pointer to the internal data array. The data array is stored in row-major order.

2.27.3.6 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE Matrix< M, N >::floatN optix::Matrix< M, N >::getRow (unsigned int m) const`

Access the specified row 0..M. Returns float, float2, float3 or float4 depending on the matrix size

2.27.3.7 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE Matrix< N, N > optix::Matrix< M, N >::identity () [static]`

Returns the identity matrix

2.27.3.8 `template<unsigned int M, unsigned int N> RT_HOSTDEVICE Matrix<4,4> optix::Matrix< M, N >::inverse () const`

Returns the inverse of the matrix

2.27.3.9 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE bool optix::Matrix< M, N >::operator< (const Matrix< M, N > & rhs) const`

Ordered comparison operator so that the matrix can be used in an STL container

2.27.3.10 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE Matrix< M, N > & optix::Matrix< M, N >::operator= (const Matrix< M, N > & b)`

Assignment operator

2.27.3.11 `template<unsigned int M, unsigned int N> RT_HOSTDEVICE float optix::Matrix< M, N >::operator[] (unsigned int i) const [inline]`

Access the specified element 0..N*M-1

2.27.3.12 `template<unsigned int M, unsigned int N> RT_HOSTDEVICE float& optix::Matrix< M, N >::operator[] (unsigned int i) [inline]`

Access the specified element 0..N*M-1

2.27.3.13 `template<unsigned int M, unsigned int N> static RT_HOSTDEVICE Matrix<4,4> optix::Matrix< M, N >::rotate (const float radians, const float3 & axis) [static]`

Returns a rotation matrix

2.27.3.14 `template<unsigned int M, unsigned int N> static RT_HOSTDEVICE Matrix<4,4> optix::Matrix< M, N >::scale (const float3 & vec) [static]`

Returns a scale matrix

2.27.3.15 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE void optix::Matrix< M, N >::setCol (unsigned int n, const floatM & c)`

Assign the specified column 0..N. Takes a float, float2, float3 or float4 depending on the matrix size

2.27.3.16 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE void optix::Matrix< M, N >::setRow (unsigned int m, const floatN & r)`

Assign the specified row 0..M. Takes a float, float2, float3 or float4 depending on the matrix size

2.27.3.17 `template<unsigned int M, unsigned int N> static RT_HOSTDEVICE Matrix<4,4> optix::Matrix< M, N >::translate (const float3 & vec) [static]`

Returns a translation matrix

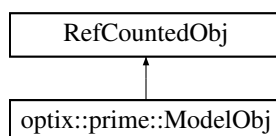
2.27.3.18 `template<unsigned int M, unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE Matrix< N, M > optix::Matrix< M, N >::transpose () const`

Returns the transpose of the matrix

2.28 optix::prime::ModelObj Class Reference

2.28.1 Detailed Description

Encapsulates an OptiX Prime model. The purpose of a model is to represent a set of triangles and an acceleration structure. Inheritance diagram for `optix::prime::ModelObj`:



Public Member Functions

- [Query createQuery](#) ([RTPquerytype](#) queryType)
- [Context getContext](#) ()
- void [finish](#) ()
- int [isFinished](#) ()
- void [update](#) (unsigned hints)
- void [copy](#) (const [Model](#) &srcModel)
- void [setTriangles](#) (RTPsize triCount, [RTPbuffertype](#) type, const void *vertPtr, unsigned stride=0)
- void [setTriangles](#) (RTPsize triCount, [RTPbuffertype](#) type, const void *indexPtr, RTPsize vertCount, [RT-Pbuffertype](#) vertType, const void *vertPtr, unsigned stride=0)
- void [setTriangles](#) (const [BufferDesc](#) &vertices)
- void [setTriangles](#) (const [BufferDesc](#) &indices, const [BufferDesc](#) &vertices)
- void [setInstances](#) (RTPsize count, [RTPbuffertype](#) instanceType, const [RTPmodel](#) *instanceList, [RT-Pbufferformat](#) transformFormat, [RTPbuffertype](#) transformType, const void *transformList)
- void [setInstances](#) (const [BufferDesc](#) &instances, const [BufferDesc](#) &transforms)
- void [setBuilderParameter](#) ([RTPbuilderparam](#) param, RTPsize size, const void *p)
- `template<typename T >`
void [setBuilderParameter](#) ([RTPbuilderparam](#) param, const T &val)
- [RTPmodel getRTPmodel](#) ()

Friends

- class [ContextObj](#)
- class [QueryObj](#)

2.28.2 Member Function Documentation

2.28.2.1 `void optix::prime::ModelObj::setBuilderParameter (RTPbuilderparam param, RTPsize size, const void * p)`
`[inline]`

Sets a model build parameter See [rtpModelSetBuilderParameter](#) for additional information

2.28.2.2 `template<typename T > void optix::prime::ModelObj::setBuilderParameter (RTPbuilderparam param, const T & val)`

Sets a model build parameter See [rtpModelSetBuilderParameter](#) for additional information

2.28.2.3 `void optix::prime::ModelObj::setInstances (RTPsize count, RTPbuffertype instanceType, const RTPmodel * instanceList, RTPbufferformat transformFormat, RTPbuffertype transformType, const void * transformList)`
`[inline]`

Sets the instance data for a model. This function creates buffer descriptors of the specified types and formats, populates them with the supplied data and assigns them to the model. See [rtpModelSetInstances](#) for additional information

2.28.2.4 `void optix::prime::ModelObj::setInstances (const BufferDesc & instances, const BufferDesc & transforms)`
`[inline]`

Sets the instance data for a model using the supplied buffer descriptors. See [rtpModelSetInstances](#) for additional information

2.28.2.5 `void optix::prime::ModelObj::setTriangles (RTPsize triCount, RTPbuffertype type, const void * vertPtr, unsigned stride = 0)` `[inline]`

Sets the triangle data for a model. This function creates a buffer descriptor of the specified type, populates it with the supplied data and assigns it to the model. The list of vertices is assumed to be a flat list of triangles and each three vertices form a single triangle. See [rtpModelSetTriangles](#) for additional information

2.28.2.6 `void optix::prime::ModelObj::setTriangles (RTPsize triCount, RTPbuffertype type, const void * indexPtr, RTPsize vertCount, RTPbuffertype vertType, const void * vertPtr, unsigned stride = 0)` `[inline]`

Sets the triangle data for a model. This function creates buffer descriptors of the specified types, populates them with the supplied data and assigns them to the model. The list of vertices uses the indices list to determine the triangles. See [rtpModelSetTriangles](#) for additional information

2.28.2.7 `void optix::prime::ModelObj::setTriangles (const BufferDesc & vertices)` `[inline]`

Sets the triangle data for a model using the supplied buffer descriptor of vertices. The list of vertices is assumed to be a flat list of triangles and each three vertices shape a single triangle. See [rtpModelSetTriangles](#) for additional information

2.28.2.8 `void optix::prime::ModelObj::setTriangles (const BufferDesc & indices, const BufferDesc & vertices)`
`[inline]`

Sets the triangle data for a model using the supplied buffer descriptor of vertices. The list of vertices uses the indices list to determine the triangles. See [rtpModelSetTriangles](#) for additional information

2.29 optix::Onb Struct Reference

2.29.1 Detailed Description

Orthonormal basis

Public Member Functions

- OPTIXU_INLINE RT_HOSTDEVICE **Onb** (const float3 &normal)
- OPTIXU_INLINE RT_HOSTDEVICE void **inverse_transform** (float3 &p) const

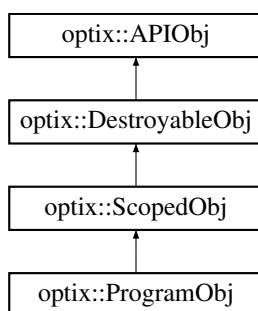
Public Attributes

- float3 **m_tangent**
- float3 **m_binormal**
- float3 **m_normal**

2.30 optix::ProgramObj Class Reference

2.30.1 Detailed Description

Program object wraps the OptiX C API RTprogram opaque type and its associated function set. Inheritance diagram for optix::ProgramObj:



Public Member Functions

- void **destroy** ()
- void **validate** ()
- **Context** **getContext** () const
- **Variable** **declareVariable** (const std::string &name)
- **Variable** **queryVariable** (const std::string &name) const
- void **removeVariable** (**Variable** v)
- unsigned int **getVariableCount** () const
- **Variable** **getVariable** (unsigned int index) const
- **RTprogram** **get** ()
- int **getId** () const

Friends

- class **Handle**< **ProgramObj** >

Additional Inherited Members

2.30.2 Member Function Documentation

2.30.2.1 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`.

2.30.2.2 `int optix::ProgramObj::getId () const [inline]`

Returns the device-side ID of this program object. See `rtProgramGetId`

2.30.2.3 `unsigned int optix::ProgramObj::getVariableCount () const [inline], [virtual]`

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

Implements `optix::ScopedObj`.

2.30.2.4 Variable `optix::ProgramObj::queryVariable (const std::string & name) const [inline], [virtual]`

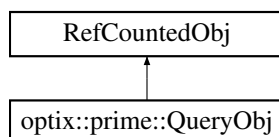
Query a variable associated with this object by name. See `rt[ObjectType]QueryVariable`. Note that this function is wrapped by the convenience function `Handle::operator[]`.

Implements `optix::ScopedObj`.

2.31 optix::prime::QueryObj Class Reference

2.31.1 Detailed Description

Encapsulates an OptiX Prime query. The purpose of a query is to coordinate the intersection of rays with a model. Inheritance diagram for `optix::prime::QueryObj`:



Public Member Functions

- `Context getContext ()`
- `void finish ()`
- `int isFinished ()`
- `void setCudaStream (cudaStream_t stream)`
- `void setRays (RTPsize count, RTPbufferformat format, RTPbuffertype type, void *rays)`
- `void setRays (const BufferDesc &rays)`
- `void setHits (RTPsize count, RTPbufferformat format, RTPbuffertype type, void *hits)`
- `void setHits (const BufferDesc &hits)`
- `void execute (unsigned hint)`
- `RTPquery getRTPquery ()`

Friends

- class `ContextObj`
- class `ModelObj`

2.32 Ray Struct Reference

2.32.1 Detailed Description

[Ray](#) class.

Description

[Ray](#) is an encapsulation of a ray mathematical entity. The origin and direction members specify the ray, while the [ray_type](#) member specifies which closest-hit/any-hit pair will be used when the ray hits a geometry object. The tmin/tmax members specify the interval over which the ray is valid.

To avoid numerical range problems, the value [RT_DEFAULT_MAX](#) can be used to specify an infinite extent.

During C++ compilation, [Ray](#) is contained within the *optix::* namespace but has global scope during C compilation. [Ray](#)'s constructors are not available during C compilation.

Members

```
// The origin of the ray
float3 origin;

// The direction of the ray
float3 direction;

// The ray type associated with this ray
unsigned int ray\_type;

// The min and max extents associated with this ray
float tmin;
float tmax;
```

Constructors

```
// Create a Ray with undefined member values
Ray( void );

// Create a Ray copied from an exemplar
Ray( const Ray &r );

// Create a ray with a specified origin, direction, ray_type, and min/max extents.
// When tmax is not given, it defaults to @ref RT_DEFAULT_MAX.
Ray( float3 origin, float3 direction, unsigned int ray_type,
    float tmin, float tmax = RT\_DEFAULT\_MAX );
```

Functions

```
// Create a ray with a specified origin, direction, ray type, and min/max extents.
Ray make_Ray( float3 origin,
    float3 direction,
    unsigned int ray_type,
    float tmin,
    float tmax );
```

History

[Ray](#) was introduced in OptiX 1.0.

See also [rtContextSetRayTypeCount](#), [rtMaterialSetAnyHitProgram](#), [rtMaterialSetClosestHitProgram](#)

Public Attributes

- float3 [origin](#)
- float3 [direction](#)
- unsigned int [ray_type](#)
- float [tmin](#)
- float [tmax](#)

2.32.2 Member Data Documentation

2.32.2.1 `float3 Ray::direction`

The direction of the ray

2.32.2.2 `float3 Ray::origin`

The origin of the ray

2.32.2.3 `unsigned int Ray::ray_type`

The ray type associated with this ray

2.32.2.4 `float Ray::tmax`

The max extent associated with this ray

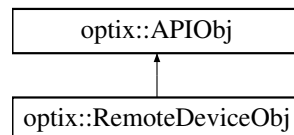
2.32.2.5 `float Ray::tmin`

The min extent associated with this ray

2.33 `optix::RemoteDeviceObj` Class Reference

2.33.1 Detailed Description

`RemoteDevice` wraps the OptiX C API `RTremotedevice` opaque type and its associated function set. Inheritance diagram for `optix::RemoteDeviceObj`:



Public Member Functions

- void **destroy** ()
- void **reserve** (unsigned int num_nodes, unsigned int configuration_idx)
- void **release** ()
- void **getAttribute** ([RTremotedeviceattribute](#) attrib, RTsize size, void *p)
- std::string **getConfiguration** (unsigned int index)
- [RTremotedevice](#) **get** ()

Static Public Member Functions

- static [RemoteDevice](#) **create** (const std::string &url, const std::string &username, const std::string &password)

Friends

- class **Handle**< `RemoteDeviceObj` >

2.34 `rtCallableProgramSizeofWrapper< T >` Struct Template Reference

Static Public Attributes

- static const `size_t value` = `sizeof(T)`

2.35 `rtCallableProgramSizeofWrapper< void >` Struct Template Reference

Static Public Attributes

- static const `size_t value` = 0

2.36 `rti_internal_typeinfo::rti_typeenum< T >` Struct Template Reference

Static Public Attributes

- static const int `m_typeenum` = `_OPTIX_TYPE_ENUM_UNKNOWN`

2.37 `rti_internal_typeinfo::rti_typeenum< optix::boundCallableProgramId< T > >` Struct Template Reference

Static Public Attributes

- static const int `m_typeenum` = `_OPTIX_TYPE_ENUM_PROGRAM_AS_ID`

2.38 `rti_internal_typeinfo::rti_typeenum< optix::callableProgramId< T > >` Struct Template Reference

Static Public Attributes

- static const int `m_typeenum` = `_OPTIX_TYPE_ENUM_PROGRAM_ID`

2.39 `rti_internal_typeinfo::rti_typeinfo` Struct Reference

Public Attributes

- unsigned int `kind`
- unsigned int `size`

2.40 `rtObject` Struct Reference**2.40.1 Detailed Description**

Opaque handle to a OptiX object.

Description

`rtObject` is an opaque handle to an OptiX object of any type. To set or query the variable value, use `rtVariableSetObject` and `rtVariableGetObject`.

Depending on how exactly the variable is used, only certain concrete types may make sense. For example, when used as an argument to `rtTrace`, the variable must be set to any OptiX type of `RTgroup`, `RTselector`, `RTgeometrygroup`, or `RTtransform`.

Note that for certain OptiX types, there are more specialized handles available to access a variable. For example, to access an OptiX object of type [RTtexturesampler](#), a handle of type [rtTextureSampler](#) provides more functionality than one of the generic type [rtObject](#).

History

[rtObject](#) was introduced in OptiX 1.0.

See also [rtVariableSetObject](#), [rtVariableGetObject](#), [rtTrace](#), [rtTextureSampler](#), [rtBuffer](#)

Protected Member Functions

- void **never_call** ()

Protected Attributes

- unsigned int **handle**

2.41 RTUtraversalresult Struct Reference

2.41.1 Detailed Description

Traversal API allowing batch raycasting queries utilizing either OptiX or the CPU.

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

Structure encapsulating the result of a single ray query

Public Attributes

- int [prim_id](#)
- float [t](#)

2.41.2 Member Data Documentation

2.41.2.1 int RTUtraversalresult::prim_id

Index of the intereseected triangle, -1 for miss

2.41.2.2 float RTUtraversalresult::t

[Ray](#) t parameter of hit point

2.42 optix::ScopedObj Class Reference

2.42.1 Detailed Description

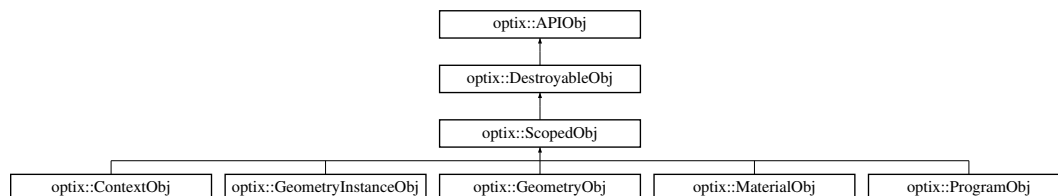
Base class for all objects which are OptiX variable containers.

Wraps:

- RTcontext
- RTgeometry
- RTgeometryinstance
- RTmaterial

- RTprogram

Inheritance diagram for optix::ScopedObj:



Public Member Functions

- virtual [Variable](#) [declareVariable](#) (const std::string &name)=0
- virtual [Variable](#) [queryVariable](#) (const std::string &name) const =0
- virtual void [removeVariable](#) ([Variable](#) v)=0
- virtual unsigned int [getVariableCount](#) () const =0
- virtual [Variable](#) [getVariable](#) (unsigned int index) const =0

Additional Inherited Members

2.42.2 Member Function Documentation

2.42.2.1 virtual [Variable](#) optix::ScopedObj::declareVariable (const std::string & *name*) [pure virtual]

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

Implemented in [optix::MaterialObj](#), [optix::GeometryObj](#), [optix::GeometryInstanceObj](#), [optix::ProgramObj](#), and [optix::ContextObj](#).

2.42.2.2 virtual unsigned int optix::ScopedObj::getVariableCount () const [pure virtual]

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

Implemented in [optix::MaterialObj](#), [optix::GeometryObj](#), [optix::GeometryInstanceObj](#), [optix::ProgramObj](#), and [optix::ContextObj](#).

2.42.2.3 virtual [Variable](#) optix::ScopedObj::queryVariable (const std::string & *name*) const [pure virtual]

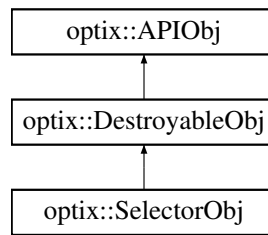
Query a variable associated with this object by name. See [rt\[ObjectType\]QueryVariable](#). Note that this function is wrapped by the convenience function [Handle::operator\[\]](#).

Implemented in [optix::MaterialObj](#), [optix::GeometryObj](#), [optix::GeometryInstanceObj](#), [optix::ProgramObj](#), and [optix::ContextObj](#).

2.43 optix::SelectorObj Class Reference

2.43.1 Detailed Description

Selector wraps the OptiX C API RTselector opaque type and its associated function set. Inheritance diagram for optix::SelectorObj:



Public Member Functions

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

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

- void `setChildCount` (unsigned int count)
- unsigned int `getChildCount` () const
- template<typename T >
void `setChild` (unsigned int index, T child)
- template<typename T >
T `getChild` (unsigned int index) const
- `RTojecttype` `getChildType` (unsigned int index) const
- template<typename T >
unsigned int `addChild` (T child)
- template<typename T >
unsigned int `removeChild` (T child)
- void `removeChild` (int index)
- void `removeChild` (unsigned int index)
- template<typename T >
unsigned int `getChildIndex` (T child) const

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

Friends

- class `Handle`< `SelectorObj` >

Additional Inherited Members

2.43.2 Member Function Documentation

2.43.2.1 template<typename T > unsigned int `optix::SelectorObj::removeChild` (T *child*) [inline]

Remove a child in this group and returns the index to the deleted element in case of success. Throws `RT_ERROR_INVALID_VALUE` if the parameter is invalid. Note: this function shifts down all the elements next to the removed one.

2.43.2.2 `void optix::SelectorObj::removeChild (int index) [inline]`

Remove a child in this group by its index. Throws `RT_ERROR_INVALID_VALUE` if the parameter is invalid. Note: this function shifts down all the elements next to the removed one.

2.43.2.3 `void optix::SelectorObj::removeChild (unsigned int index) [inline]`

Remove a child in this group by its index. Throws `RT_ERROR_INVALID_VALUE` if the parameter is invalid. Note: this function shifts down all the elements next to the removed one.

2.43.2.4 `void optix::SelectorObj::setChildCount (unsigned int count) [inline]`

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

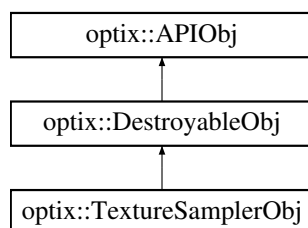
2.43.2.5 `void optix::SelectorObj::setVisitProgram (Program program) [inline]`

Set the visitor program for this selector. See `rtSelectorSetVisitProgram`

2.44 `optix::TextureSamplerObj` Class Reference

2.44.1 Detailed Description

TextureSampler wraps the OptiX C API `RTtexturesampler` opaque type and its associated function set. Inheritance diagram for `optix::TextureSamplerObj`:



Public Member Functions

- `void destroy ()`
- `void validate ()`
- `Context getContext () const`
- `RTtexturesampler get ()`
- `void setMipLevelCount (unsigned int num_mip_levels)`
- `unsigned int getMipLevelCount () const`
- `void setArraySize (unsigned int num_textures_in_array)`
- `unsigned int getArraySize () const`
- `void setWrapMode (unsigned int dim, RTwrapmode wrapmode)`
- `RTwrapmode getWrapMode (unsigned int dim) const`
- `void setFilteringModes (RTfiltermode minification, RTfiltermode magnification, RTfiltermode mipmapping)`
- `void getFilteringModes (RTfiltermode &minification, RTfiltermode &magnification, RTfiltermode &mipmapping) const`
- `void setMaxAnisotropy (float value)`
- `float getMaxAnisotropy () const`
- `void setMipLevelClamp (float minLevel, float maxLevel)`
- `void getMipLevelClamp (float &minLevel, float &maxLevel) const`
- `void setMipLevelBias (float value)`
- `float getMipLevelBias () const`

- void [setReadMode](#) ([RTtexturereadmode](#) readmode)
- [RTtexturereadmode](#) [getReadMode](#) () const
- void [setIndexingMode](#) ([RTtextureindexmode](#) indexmode)
- [RTtextureindexmode](#) [getIndexingMode](#) () const
- int [getId](#) () const
- void [setBuffer](#) (unsigned int texture_array_idx, unsigned int mip_level, [Buffer](#) buffer)
- [Buffer](#) [getBuffer](#) (unsigned int texture_array_idx, unsigned int mip_level) const
- void [setBuffer](#) ([Buffer](#) buffer)
- [Buffer](#) [getBuffer](#) () const
- void [registerGLTexture](#) ()
- void [unregisterGLTexture](#) ()

Friends

- class **Handle**< **TextureSamplerObj** >

Additional Inherited Members

2.44.2 Member Function Documentation

2.44.2.1 int optix::TextureSamplerObj::getId () const [inline]

Returns the device-side ID of this sampler. See [rtTextureSamplerGetId](#)

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

Declare the texture's buffer as immutable and accessible by OptiX. See [rtTextureSamplerGLRegister](#).

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

Deprecated in OptiX 4.0 Set the underlying buffer used for texture storage. See [rtTextureSamplerSetBuffer](#).

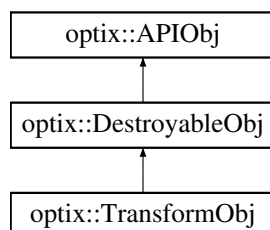
2.44.2.4 void optix::TextureSamplerObj::setMipLevelCount (unsigned int *num_mip_levels*) [inline]

Deprecated in OptiX 4.0 Set the number of mip levels for this sampler. See [rtTextureSamplerSetMipLevelCount](#).

2.45 optix::TransformObj Class Reference

2.45.1 Detailed Description

Transform wraps the OptiX C API RTtransform opaque type and its associated function set. Inheritance diagram for optix::TransformObj:



Public Member Functions

- void `destroy` ()
- void `validate` ()
- `Context` `getContext` () const
- `RTtransform` `get` ()
- template<typename T >
void `setChild` (T child)
- template<typename T >
T `getChild` () const
- `RObjectType` `getChildType` () const
- void `setMatrix` (bool transpose, const float *matrix, const float *inverse_matrix)
- void `getMatrix` (bool transpose, float *matrix, float *inverse_matrix) const

Friends

- class `Handle< TransformObj >`

Additional Inherited Members

2.45.2 Member Function Documentation

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

Set the child node of this transform. See [rtTransformSetChild](#).

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

Set the transform matrix for this node. See [rtTransformSetMatrix](#).

2.46 `optix::buffer< T, Dim >::type< T2 >` Struct Template Reference2.47 `optix::VariableObj` Class Reference

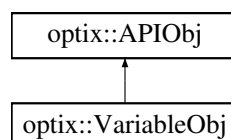
2.47.1 Detailed Description

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

See [OptiXApiReference](#) 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. Inheritance diagram for `optix::VariableObj`:



Public Member Functions

- `Context` `getContext ()` const
- `std::string` `getName ()` const
- `std::string` `getAnnotation ()` const
- `RObjectType` `getType ()` const
- `RTvariable` `get ()`
- `RTsize` `getSize ()` const

Float setters

Set variable to have a float value.

- void `setFloat` (float f1)
- void `setFloat` (optix::float2 f)
- void `setFloat` (float f1, float f2)
- void `setFloat` (optix::float3 f)
- void `setFloat` (float f1, float f2, float f3)
- void `setFloat` (optix::float4 f)
- void `setFloat` (float f1, float f2, float f3, float f4)
- void `set1fv` (const float *f)
- void `set2fv` (const float *f)
- void `set3fv` (const float *f)
- void `set4fv` (const float *f)

Int setters

Set variable to have an int value.

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

Unsigned int setters

Set variable to have an unsigned int value.

- void `setUInt` (unsigned int u1)
- void `setUInt` (unsigned int u1, unsigned int u2)
- void `setUInt` (unsigned int u1, unsigned int u2, unsigned int u3)
- void `setUInt` (unsigned int u1, unsigned int u2, unsigned int u3, unsigned int u4)
- void `setUInt` (optix::uint2 u)
- void `setUInt` (optix::uint3 u)
- void `setUInt` (optix::uint4 u)
- void `set1uiv` (const unsigned int *u)
- void `set2uiv` (const unsigned int *u)
- void `set3uiv` (const unsigned int *u)
- void `set4uiv` (const unsigned int *u)

Matrix setters

Set variable to have a `Matrix` value

- void `setMatrix2x2fv` (bool transpose, const float *m)
- void `setMatrix2x3fv` (bool transpose, const float *m)
- void `setMatrix2x4fv` (bool transpose, const float *m)

- void **setMatrix3x2fv** (bool transpose, const float *m)
- void **setMatrix3x3fv** (bool transpose, const float *m)
- void **setMatrix3x4fv** (bool transpose, const float *m)
- void **setMatrix4x2fv** (bool transpose, const float *m)
- void **setMatrix4x3fv** (bool transpose, const float *m)
- void **setMatrix4x4fv** (bool transpose, const float *m)

Numeric value getters

Query value of a variable with numeric value

- float **getFloat** () const
- `optix::float2` **getFloat2** () const
- `optix::float3` **getFloat3** () const
- `optix::float4` **getFloat4** () const
- void **getFloat** (float &f1) const
- void **getFloat** (float &f1, float &f2) const
- void **getFloat** (float &f1, float &f2, float &f3) const
- void **getFloat** (float &f1, float &f2, float &f3, float &f4) const
- unsigned **getUInt** () const
- `optix::uint2` **getUInt2** () const
- `optix::uint3` **getUInt3** () const
- `optix::uint4` **getUInt4** () const
- void **getUInt** (unsigned &u1) const
- void **getUInt** (unsigned &u1, unsigned &u2) const
- void **getUInt** (unsigned &u1, unsigned &u2, unsigned &u3) const
- void **getUInt** (unsigned &u1, unsigned &u2, unsigned &u3, unsigned &u4) const
- int **getInt** () const
- `optix::int2` **getInt2** () const
- `optix::int3` **getInt3** () const
- `optix::int4` **getInt4** () const
- void **getInt** (int &i1) const
- void **getInt** (int &i1, int &i2) const
- void **getInt** (int &i1, int &i2, int &i3) const
- void **getInt** (int &i1, int &i2, int &i3, int &i4) const
- void **getMatrix2x2** (bool transpose, float *m) const
- void **getMatrix2x3** (bool transpose, float *m) const
- void **getMatrix2x4** (bool transpose, float *m) const
- void **getMatrix3x2** (bool transpose, float *m) const
- void **getMatrix3x3** (bool transpose, float *m) const
- void **getMatrix3x4** (bool transpose, float *m) const
- void **getMatrix4x2** (bool transpose, float *m) const
- void **getMatrix4x3** (bool transpose, float *m) const
- void **getMatrix4x4** (bool transpose, float *m) const

OptiX API object setters

Set variable to have an OptiX API object as its value

- void **setBuffer** ([Buffer](#) buffer)
- void **set** ([Buffer](#) buffer)
- void **setTextureSampler** ([TextureSampler](#) texturesample)
- void **set** ([TextureSampler](#) texturesample)
- void **set** ([GeometryGroup](#) group)
- void **set** ([Group](#) group)
- void **set** ([Program](#) program)
- void **setProgramId** ([Program](#) program)
- void **set** ([Selector](#) selector)
- void **set** ([Transform](#) transform)

OptiX API object getters

Retrieve OptiX API object value from a variable

- [Buffer](#) **getBuffer** () const

- [GeometryGroup](#) **getGeometryGroup** () const
- [GeometryInstance](#) **getGeometryInstance** () const
- [Group](#) **getGroup** () const
- [Program](#) **getProgram** () const
- [Selector](#) **getSelector** () const
- [TextureSampler](#) **getTextureSampler** () const
- [Transform](#) **getTransform** () const

User data variable accessors

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

Friends

- class **Handle**< **VariableObj** >

Additional Inherited Members

2.48 optix::VectorDim< DIM > Struct Template Reference

2.49 optix::VectorDim< 2 > Struct Template Reference

Public Types

- typedef float2 **VectorType**

2.50 optix::VectorDim< 3 > Struct Template Reference

Public Types

- typedef float3 **VectorType**

2.51 optix::VectorDim< 4 > Struct Template Reference

Public Types

- typedef float4 **VectorType**

2.52 optix::VectorTypes< T, Dim > Struct Template Reference

2.53 optix::VectorTypes< float, 1 > Struct Template Reference

Public Types

- typedef float **Type**

Static Public Member Functions

- template<class S >
static __device__
__forceinline__ Type **make** (S s)

2.54 `optix::VectorTypes< float, 2 >` Struct Template Reference

Public Types

- typedef float2 **Type**

Static Public Member Functions

- template<class S >
static __device__
__forceinline__ Type **make** (S s)

2.55 `optix::VectorTypes< float, 3 >` Struct Template Reference

Public Types

- typedef float3 **Type**

Static Public Member Functions

- template<class S >
static __device__
__forceinline__ Type **make** (S s)

2.56 `optix::VectorTypes< float, 4 >` Struct Template Reference

Public Types

- typedef float4 **Type**

Static Public Member Functions

- template<class S >
static __device__
__forceinline__ Type **make** (S s)

2.57 `optix::VectorTypes< int, 1 >` Struct Template Reference

Public Types

- typedef int **Type**

Static Public Member Functions

- template<class S >
static __device__
__forceinline__ Type **make** (S s)

2.58 `optix::VectorTypes< int, 2 >` Struct Template Reference

Public Types

- typedef int2 **Type**

Static Public Member Functions

- template<class S >
static __device__
__forceinline__ Type **make** (S s)

2.59 optix::VectorTypes< int, 3 > Struct Template Reference**Public Types**

- typedef int3 **Type**

Static Public Member Functions

- template<class S >
static __device__
__forceinline__ Type **make** (S s)

2.60 optix::VectorTypes< int, 4 > Struct Template Reference**Public Types**

- typedef int4 **Type**

Static Public Member Functions

- template<class S >
static __device__
__forceinline__ Type **make** (S s)

2.61 optix::VectorTypes< unsigned int, 1 > Struct Template Reference**Public Types**

- typedef unsigned int **Type**

Static Public Member Functions

- static __device__
__forceinline__ Type **make** (unsigned int s)
- template<class S >
static __device__
__forceinline__ Type **make** (S s)

2.62 optix::VectorTypes< unsigned int, 2 > Struct Template Reference**Public Types**

- typedef uint2 **Type**

Static Public Member Functions

- `template<class S >`
`static __device__`
`__forceinline__ Type make (S s)`

2.63 `optix::VectorTypes< unsigned int, 3 >` Struct Template Reference**Public Types**

- `typedef uint3 Type`

Static Public Member Functions

- `template<class S >`
`static __device__`
`__forceinline__ Type make (S s)`

2.64 `optix::VectorTypes< unsigned int, 4 >` Struct Template Reference**Public Types**

- `typedef uint4 Type`

Static Public Member Functions

- `template<class S >`
`static __device__`
`__forceinline__ Type make (S s)`

3 File Documentation

3.1 `optix.h` File Reference

3.1.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation Includes the host api if compiling host code, includes the cuda api if compiling device code. For the math library routines include [optix_math.h](#)

Macros

- `#define OPTIX_VERSION`

3.1.2 Macro Definition Documentation

3.1.2.1 `#define OPTIX_VERSION`

Value:

```

40000 /* major = OPTIX_VERSION/10000,          *
      * minor = (OPTIX_VERSION%10000)/100,      *
      * micro = OPTIX_VERSION%100              */

```

3.2 optix_cuda_interop.h File Reference

3.2.1 Detailed Description

OptiX public API declarations CUDAInterop.

Author

NVIDIA Corporation OptiX public API declarations for CUDA interoperability

Functions

- [RTresult](#) RTAPI [rtBufferCreateForCUDA](#) ([RTcontext](#) context, unsigned int bufferdesc, [RTbuffer](#) *buffer)
- [RTresult](#) RTAPI [rtBufferGetDevicePointer](#) ([RTbuffer](#) buffer, int optix_device_ordinal, void **device_pointer)
- [RTresult](#) RTAPI [rtBufferMarkDirty](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtBufferSetDevicePointer](#) ([RTbuffer](#) buffer, int optix_device_ordinal, void *device_pointer)

3.2.2 Function Documentation

3.2.2.1 RTresult RTAPI rtBufferCreateForCUDA (RTcontext context, unsigned int bufferdesc, RTbuffer * buffer)

Creates a new buffer object that will later rely on user-side CUDA allocation.

Description

DEPRECATED in OptiX 4.0. Now forwards to [rtBufferCreate](#).

Parameters

| | | |
|-----|-------------------|---|
| in | <i>context</i> | The context to create the buffer in |
| in | <i>bufferdesc</i> | Bitwise <i>or</i> combination of the <i>type</i> and <i>flags</i> of the new buffer |
| out | <i>buffer</i> | The return handle for the buffer object |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferCreateForCUDA](#) was introduced in OptiX 3.0.

See also [rtBufferCreate](#), [rtBufferSetDevicePointer](#), [rtBufferMarkDirty](#), [rtBufferDestroy](#)

3.2.2.2 RTresult RTAPI rtBufferGetDevicePointer (RTbuffer buffer, int optix_device_ordinal, void ** device_pointer)

Gets the pointer to the buffer's data on the given device.

Description

[rtBufferGetDevicePointer](#) returns the pointer to the data of *buffer* on device *optix_device_ordinal* in ***device_pointer*.

If [rtBufferGetDevicePointer](#) has been called for a single device for a given buffer, the user can change the buffer's content on that device through the pointer. OptiX must then synchronize the new buffer contents to all devices. These synchronization copies occur at every `rtContextLaunch`, unless the buffer is created with [RT_BUFFER_COPY_ON_DIRTY](#). In this case, [rtBufferMarkDirty](#) can be used to notify OptiX that the buffer has been dirtied and must be synchronized.

Parameters

| | | |
|-----|-----------------------------|--|
| in | <i>buffer</i> | The buffer to be queried for its device pointer |
| in | <i>optix_device_ordinal</i> | The number assigned by OptiX to the device |
| out | <i>device_pointer</i> | The return handle to the buffer's device pointer |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtBufferGetDevicePointer](#) was introduced in OptiX 3.0.

See also [rtBufferMarkDirty](#), [rtBufferSetDevicePointer](#)

3.2.2.3 RTresult RTAPI rtBufferMarkDirty (RTbuffer buffer)

Sets a buffer as dirty.

Description

If [rtBufferSetDevicePointer](#) or [rtBufferGetDevicePointer](#) have been called for a single device for a given buffer, the user can change the buffer's content on that device through the pointer. OptiX must then synchronize the new buffer contents to all devices. These synchronization copies occur at every `rtContextLaunch`, unless the buffer is declared with [RT_BUFFER_COPY_ON_DIRTY](#). In this case, [rtBufferMarkDirty](#) can be used to notify OptiX that the buffer has been dirtied and must be synchronized.

Note that `RT_BUFFER_COPY_ON_DIRTY` currently only applies to CUDA interop buffers (buffers for which the application has a device pointer).

Parameters

| | | |
|----|---------------|-------------------------------|
| in | <i>buffer</i> | The buffer to be marked dirty |
|----|---------------|-------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtBufferMarkDirty](#) was introduced in OptiX 3.0.

See also [rtBufferGetDevicePointer](#), [rtBufferSetDevicePointer](#), [RT_BUFFER_COPY_ON_DIRTY](#)

3.2.2.4 RTresult RTAPI rtBufferSetDevicePointer (RTbuffer buffer, int optix_device_ordinal, void * device_pointer)

Sets the pointer to the buffer's data on the given device.

Description

[rtBufferSetDevicePointer](#) sets the pointer to the data of *buffer* on device *optix_device_ordinal* to *device_pointer*.

If [rtBufferSetDevicePointer](#) has been called for a single device for a given buffer, the user can change the buffer's content on that device through the pointer. OptiX must then synchronize the new buffer contents to all devices. These synchronization copies occur at every `rtContextLaunch`, unless the buffer is declared with [RT_BUFFER_COPY_ON_DIRTY](#). In this case, [rtBufferMarkDirty](#) can be used to notify OptiX that the buffer has been dirtied and must be synchronized.

Parameters

| | | |
|----|-----------------------------|--|
| in | <i>buffer</i> | The buffer for which the device pointer is to be set |
| in | <i>optix_device_ordinal</i> | The number assigned by OptiX to the device |
| in | <i>device_pointer</i> | The pointer to the data on the specified device |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_INVALID_CONTEXT](#)

History

[rtBufferSetDevicePointer](#) was introduced in OptiX 3.0.

See also [rtBufferMarkDirty](#), [rtBufferGetDevicePointer](#)

3.3 optix_datatypes.h File Reference

3.3.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Datatypes

Classes

- struct [Ray](#)

Macros

- `#define` [RT_DEFAULT_MAX](#) 1.e27f

Functions

- static `__inline__` `RT_HOSTDEVICE` [Ray](#) **make_Ray** (float3 origin, float3 direction, unsigned int ray_type, float tmin, float tmax)

3.3.2 Macro Definition Documentation

3.3.2.1 `#define` [RT_DEFAULT_MAX](#) 1.e27f

Max t for a ray

3.4 optix_declarations.h File Reference

3.4.1 Detailed Description

OptiX public API declarations.

Author

NVIDIA Corporation OptiX public API declarations

Enumerations

- enum `RTformat` {
`RT_FORMAT_UNKNOWN` = 0x100,
`RT_FORMAT_FLOAT`,
`RT_FORMAT_FLOAT2`,
`RT_FORMAT_FLOAT3`,
`RT_FORMAT_FLOAT4`,
`RT_FORMAT_BYTE`,
`RT_FORMAT_BYTE2`,
`RT_FORMAT_BYTE3`,
`RT_FORMAT_BYTE4`,
`RT_FORMAT_UNSIGNED_BYTE`,
`RT_FORMAT_UNSIGNED_BYTE2`,
`RT_FORMAT_UNSIGNED_BYTE3`,
`RT_FORMAT_UNSIGNED_BYTE4`,
`RT_FORMAT_SHORT`,
`RT_FORMAT_SHORT2`,
`RT_FORMAT_SHORT3`,
`RT_FORMAT_SHORT4`,
`RT_FORMAT_UNSIGNED_SHORT`,
`RT_FORMAT_UNSIGNED_SHORT2`,
`RT_FORMAT_UNSIGNED_SHORT3`,
`RT_FORMAT_UNSIGNED_SHORT4`,
`RT_FORMAT_INT`,
`RT_FORMAT_INT2`,
`RT_FORMAT_INT3`,
`RT_FORMAT_INT4`,
`RT_FORMAT_UNSIGNED_INT`,
`RT_FORMAT_UNSIGNED_INT2`,
`RT_FORMAT_UNSIGNED_INT3`,
`RT_FORMAT_UNSIGNED_INT4`,
`RT_FORMAT_USER`,
`RT_FORMAT_BUFFER_ID`,
`RT_FORMAT_PROGRAM_ID`,
`RT_FORMAT_HALF`,
`RT_FORMAT_HALF2`,
`RT_FORMAT_HALF3`,
`RT_FORMAT_HALF4` }
- enum `RTobjecttype` {

```

RT_OBJECTTYPE_UNKNOWN = 0x200,
RT_OBJECTTYPE_GROUP,
RT_OBJECTTYPE_GEOMETRY_GROUP,
RT_OBJECTTYPE_TRANSFORM,
RT_OBJECTTYPE_SELECTOR,
RT_OBJECTTYPE_GEOMETRY_INSTANCE,
RT_OBJECTTYPE_BUFFER,
RT_OBJECTTYPE_TEXTURE_SAMPLER,
RT_OBJECTTYPE_OBJECT,
RT_OBJECTTYPE_MATRIX_FLOAT2x2,
RT_OBJECTTYPE_MATRIX_FLOAT2x3,
RT_OBJECTTYPE_MATRIX_FLOAT2x4,
RT_OBJECTTYPE_MATRIX_FLOAT3x2,
RT_OBJECTTYPE_MATRIX_FLOAT3x3,
RT_OBJECTTYPE_MATRIX_FLOAT3x4,
RT_OBJECTTYPE_MATRIX_FLOAT4x2,
RT_OBJECTTYPE_MATRIX_FLOAT4x3,
RT_OBJECTTYPE_MATRIX_FLOAT4x4,
RT_OBJECTTYPE_FLOAT,
RT_OBJECTTYPE_FLOAT2,
RT_OBJECTTYPE_FLOAT3,
RT_OBJECTTYPE_FLOAT4,
RT_OBJECTTYPE_INT,
RT_OBJECTTYPE_INT2,
RT_OBJECTTYPE_INT3,
RT_OBJECTTYPE_INT4,
RT_OBJECTTYPE_UNSIGNED_INT,
RT_OBJECTTYPE_UNSIGNED_INT2,
RT_OBJECTTYPE_UNSIGNED_INT3,
RT_OBJECTTYPE_UNSIGNED_INT4,
RT_OBJECTTYPE_USER,
RT_OBJECTTYPE_PROGRAM }

• enum RTwrapmode {
    RT_WRAP_REPEAT,
    RT_WRAP_CLAMP_TO_EDGE,
    RT_WRAP_MIRROR,
    RT_WRAP_CLAMP_TO_BORDER }

• enum RTfiltermode {
    RT_FILTER_NEAREST,
    RT_FILTER_LINEAR,
    RT_FILTER_NONE }

• enum RTtexturereadmode {
    RT_TEXTURE_READ_ELEMENT_TYPE = 0,
    RT_TEXTURE_READ_NORMALIZED_FLOAT = 1,
    RT_TEXTURE_READ_ELEMENT_TYPE_SRGB = 2,
    RT_TEXTURE_READ_NORMALIZED_FLOAT_SRGB = 3 }

• enum RTgltarget {
    RT_TARGET_GL_TEXTURE_2D,
    RT_TARGET_GL_TEXTURE_RECTANGLE,
    RT_TARGET_GL_TEXTURE_3D,
    RT_TARGET_GL_RENDER_BUFFER,
    RT_TARGET_GL_TEXTURE_1D,
    RT_TARGET_GL_TEXTURE_1D_ARRAY,
    RT_TARGET_GL_TEXTURE_2D_ARRAY,
    RT_TARGET_GL_TEXTURE_CUBE_MAP,
    RT_TARGET_GL_TEXTURE_CUBE_MAP_ARRAY }

• enum RTtextureindexmode {
    RT_TEXTURE_INDEX_NORMALIZED_COORDINATES,

```

- `RT_TEXTURE_INDEX_ARRAY_INDEX }`
- `enum RTbuffertype {`
 - `RT_BUFFER_INPUT = 0x1,`
 - `RT_BUFFER_OUTPUT = 0x2,`
 - `RT_BUFFER_INPUT_OUTPUT = RT_BUFFER_INPUT | RT_BUFFER_OUTPUT,`
 - `RT_BUFFER_PROGRESSIVE_STREAM = 0x10 }`
- `enum RTbufferflag {`
 - `RT_BUFFER_GPU_LOCAL = 0x4,`
 - `RT_BUFFER_COPY_ON_DIRTY = 0x8,`
 - `RT_BUFFER_LAYERED = 0x200000,`
 - `RT_BUFFER_CUBEMAP = 0x400000 }`
- `enum RTbuffermapflag {`
 - `RT_BUFFER_MAP_READ = 0x1,`
 - `RT_BUFFER_MAP_READ_WRITE = 0x2,`
 - `RT_BUFFER_MAP_WRITE = 0x4,`
 - `RT_BUFFER_MAP_WRITE_DISCARD = 0x8 }`
- `enum RTexception {`
 - `RT_EXCEPTION_PROGRAM_ID_INVALID = 0x3EE,`
 - `RT_EXCEPTION_TEXTURE_ID_INVALID = 0x3EF,`
 - `RT_EXCEPTION_BUFFER_ID_INVALID = 0x3FA,`
 - `RT_EXCEPTION_INDEX_OUT_OF_BOUNDS = 0x3FB,`
 - `RT_EXCEPTION_STACK_OVERFLOW = 0x3FC,`
 - `RT_EXCEPTION_BUFFER_INDEX_OUT_OF_BOUNDS = 0x3FD,`
 - `RT_EXCEPTION_INVALID_RAY = 0x3FE,`
 - `RT_EXCEPTION_INTERNAL_ERROR = 0x3FF,`
 - `RT_EXCEPTION_USER = 0x400,`
 - `RT_EXCEPTION_ALL = 0x7FFFFFFF }`
- `enum RResult {`

```

RT_SUCCESS = 0,
RT_TIMEOUT_CALLBACK = 0x100,
RT_ERROR_INVALID_CONTEXT = 0x500,
RT_ERROR_INVALID_VALUE = 0x501,
RT_ERROR_MEMORY_ALLOCATION_FAILED = 0x502,
RT_ERROR_TYPE_MISMATCH = 0x503,
RT_ERROR_VARIABLE_NOT_FOUND = 0x504,
RT_ERROR_VARIABLE_REDECLARED = 0x505,
RT_ERROR_ILLEGAL_SYMBOL = 0x506,
RT_ERROR_INVALID_SOURCE = 0x507,
RT_ERROR_VERSION_MISMATCH = 0x508,
RT_ERROR_OBJECT_CREATION_FAILED = 0x600,
RT_ERROR_NO_DEVICE = 0x601,
RT_ERROR_INVALID_DEVICE = 0x602,
RT_ERROR_INVALID_IMAGE = 0x603,
RT_ERROR_FILE_NOT_FOUND = 0x604,
RT_ERROR_ALREADY_MAPPED = 0x605,
RT_ERROR_INVALID_DRIVER_VERSION = 0x606,
RT_ERROR_CONTEXT_CREATION_FAILED = 0x607,
RT_ERROR_RESOURCE_NOT_REGISTERED = 0x608,
RT_ERROR_RESOURCE_ALREADY_REGISTERED = 0x609,
RT_ERROR_LAUNCH_FAILED = 0x900,
RT_ERROR_NOT_SUPPORTED = 0xA00,
RT_ERROR_CONNECTION_FAILED = 0xB00,
RT_ERROR_AUTHENTICATION_FAILED = 0xB01,
RT_ERROR_CONNECTION_ALREADY_EXISTS = 0xB02,
RT_ERROR_NETWORK_LOAD_FAILED = 0xB03,
RT_ERROR_NETWORK_INIT_FAILED = 0xB04,
RT_ERROR_CLUSTER_NOT_RUNNING = 0xB06,
RT_ERROR_CLUSTER_ALREADY_RUNNING = 0xB07,
RT_ERROR_INSUFFICIENT_FREE_NODES = 0xB08,
RT_ERROR_UNKNOWN = ~0 }

• enum RTdeviceattribute {
    RT_DEVICE_ATTRIBUTE_MAX_THREADS_PER_BLOCK,
    RT_DEVICE_ATTRIBUTE_CLOCK_RATE,
    RT_DEVICE_ATTRIBUTE_MULTIPROCESSOR_COUNT,
    RT_DEVICE_ATTRIBUTE_EXECUTION_TIMEOUT_ENABLED,
    RT_DEVICE_ATTRIBUTE_MAX_HARDWARE_TEXTURE_COUNT,
    RT_DEVICE_ATTRIBUTE_NAME,
    RT_DEVICE_ATTRIBUTE_COMPUTE_CAPABILITY,
    RT_DEVICE_ATTRIBUTE_TOTAL_MEMORY,
    RT_DEVICE_ATTRIBUTE_TCC_DRIVER,
    RT_DEVICE_ATTRIBUTE_CUDA_DEVICE_ORDINAL }

• enum RTremotedeviceattribute {
    RT_REMOTEDEVICE_ATTRIBUTE_CLUSTER_URL,
    RT_REMOTEDEVICE_ATTRIBUTE_HEAD_NODE_URL,
    RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS,
    RT_REMOTEDEVICE_ATTRIBUTE_STATUS,
    RT_REMOTEDEVICE_ATTRIBUTE_NUM_TOTAL_NODES,
    RT_REMOTEDEVICE_ATTRIBUTE_NUM_FREE_NODES,
    RT_REMOTEDEVICE_ATTRIBUTE_NUM_RESERVED_NODES,
    RT_REMOTEDEVICE_ATTRIBUTE_NAME,
    RT_REMOTEDEVICE_ATTRIBUTE_NUM_GPUS,
    RT_REMOTEDEVICE_ATTRIBUTE_GPU_TOTAL_MEMORY,
    RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS = 0x04000000 }

• enum RTremotedevicestatus {

```

- RT_REMOTEDEVICE_STATUS_READY,
- RT_REMOTEDEVICE_STATUS_CONNECTED,
- RT_REMOTEDEVICE_STATUS_RESERVED,
- RT_REMOTEDEVICE_STATUS_DISCONNECTED = ~0 }
- enum RTcontextattribute {
 - RT_CONTEXT_ATTRIBUTE_MAX_TEXTURE_COUNT,
 - RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS,
 - RT_CONTEXT_ATTRIBUTE_USED_HOST_MEMORY,
 - RT_CONTEXT_ATTRIBUTE_GPU_PAGING_ACTIVE,
 - RT_CONTEXT_ATTRIBUTE_GPU_PAGING_FORCED_OFF,
 - RT_CONTEXT_ATTRIBUTE_AVAILABLE_DEVICE_MEMORY = 0x10000000 }
- enum RTbufferattribute {
 - RT_BUFFER_ATTRIBUTE_STREAM_FORMAT,
 - RT_BUFFER_ATTRIBUTE_STREAM_BITRATE,
 - RT_BUFFER_ATTRIBUTE_STREAM_FPS,
 - RT_BUFFER_ATTRIBUTE_STREAM_GAMMA }
- enum RTbufferidnull { RT_BUFFER_ID_NULL = 0 }
- enum RTprogramidnull { RT_PROGRAM_ID_NULL = 0 }
- enum RTtextureidnull { RT_TEXTURE_ID_NULL = 0 }

3.4.2 Enumeration Type Documentation

3.4.2.1 enum RTbufferattribute

Buffer attributes

Enumerator

- RT_BUFFER_ATTRIBUTE_STREAM_FORMAT** Format string
- RT_BUFFER_ATTRIBUTE_STREAM_BITRATE** sizeof(int)
- RT_BUFFER_ATTRIBUTE_STREAM_FPS** sizeof(int)
- RT_BUFFER_ATTRIBUTE_STREAM_GAMMA** sizeof(float)

3.4.2.2 enum RTbufferflag

Buffer flags

Enumerator

- RT_BUFFER_GPU_LOCAL** An [RT_BUFFER_INPUT_OUTPUT](#) has separate copies on each device that are not synchronized
- RT_BUFFER_COPY_ON_DIRTY** A CUDA Interop buffer will only be synchronized across devices when dirtied by [rtBufferMap](#) or [rtBufferMarkDirty](#)
- RT_BUFFER_LAYERED** Depth specifies the number of layers, not the depth of a 3D array
- RT_BUFFER_CUBEMAP** Enables creation of cubemaps. If this flag is set, Width must be equal to Height, and Depth must be six. If the [RT_BUFFER_LAYERED](#) flag is also set, then Depth must be a multiple of six

3.4.2.3 enum RTbufferidnull

Sentinel values

Enumerator

- RT_BUFFER_ID_NULL** sentinel for describing a non-existent buffer id

3.4.2.4 enum RTbuffermapflag

Buffer mapping flags

Enumerator

RT_BUFFER_MAP_READ Map buffer memory for reading
RT_BUFFER_MAP_READ_WRITE Map buffer memory for both reading and writing
RT_BUFFER_MAP_WRITE Map buffer memory for writing
RT_BUFFER_MAP_WRITE_DISCARD Map buffer memory for writing, with the previous contents being undefined

3.4.2.5 enum RTbuffertype

Buffer type

Enumerator

RT_BUFFER_INPUT Input buffer for the GPU
RT_BUFFER_OUTPUT Output buffer for the GPU
RT_BUFFER_INPUT_OUTPUT Output/Input buffer for the GPU
RT_BUFFER_PROGRESSIVE_STREAM Progressive stream buffer

3.4.2.6 enum RTcontextattribute

Context attributes

Enumerator

RT_CONTEXT_ATTRIBUTE_MAX_TEXTURE_COUNT sizeof(int)
RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS sizeof(int)
RT_CONTEXT_ATTRIBUTE_USED_HOST_MEMORY sizeof(RTsize)
RT_CONTEXT_ATTRIBUTE_GPU_PAGING_ACTIVE sizeof(int)
RT_CONTEXT_ATTRIBUTE_GPU_PAGING_FORCED_OFF sizeof(int)
RT_CONTEXT_ATTRIBUTE_AVAILABLE_DEVICE_MEMORY sizeof(RTsize)

3.4.2.7 enum RTdeviceattribute

Device attributes

Enumerator

RT_DEVICE_ATTRIBUTE_MAX_THREADS_PER_BLOCK Max Threads per Block
RT_DEVICE_ATTRIBUTE_CLOCK_RATE Clock rate
RT_DEVICE_ATTRIBUTE_MULTIPROCESSOR_COUNT Multiprocessor count
RT_DEVICE_ATTRIBUTE_EXECUTION_TIMEOUT_ENABLED Execution timeout enabled
RT_DEVICE_ATTRIBUTE_MAX_HARDWARE_TEXTURE_COUNT Hardware Texture count
RT_DEVICE_ATTRIBUTE_NAME Attribute Name
RT_DEVICE_ATTRIBUTE_COMPUTE_CAPABILITY Compute Capabilities
RT_DEVICE_ATTRIBUTE_TOTAL_MEMORY Total Memory
RT_DEVICE_ATTRIBUTE_TCC_DRIVER sizeof(int)
RT_DEVICE_ATTRIBUTE_CUDA_DEVICE_ORDINAL sizeof(int)

3.4.2.8 enum RTexception

Exceptions

Enumerator

RT_EXCEPTION_PROGRAM_ID_INVALID Program ID not valid
RT_EXCEPTION_TEXTURE_ID_INVALID Texture ID not valid
RT_EXCEPTION_BUFFER_ID_INVALID Buffer ID not valid
RT_EXCEPTION_INDEX_OUT_OF_BOUNDS Index out of bounds
RT_EXCEPTION_STACK_OVERFLOW Stack overflow
RT_EXCEPTION_BUFFER_INDEX_OUT_OF_BOUNDS Buffer index out of bounds
RT_EXCEPTION_INVALID_RAY Invalid ray
RT_EXCEPTION_INTERNAL_ERROR Internal error
RT_EXCEPTION_USER User exception
RT_EXCEPTION_ALL All exceptions

3.4.2.9 enum RTfiltermode

Filter mode

Enumerator

RT_FILTER_NEAREST Nearest
RT_FILTER_LINEAR Linear
RT_FILTER_NONE No filter

3.4.2.10 enum RTformat

OptiX formats

Enumerator

RT_FORMAT_UNKNOWN Format unknown
RT_FORMAT_FLOAT Float
RT_FORMAT_FLOAT2 sizeof(float)*2
RT_FORMAT_FLOAT3 sizeof(float)*3
RT_FORMAT_FLOAT4 sizeof(float)*4
RT_FORMAT_BYTE BYTE
RT_FORMAT_BYTE2 sizeof(CHAR)*2
RT_FORMAT_BYTE3 sizeof(CHAR)*3
RT_FORMAT_BYTE4 sizeof(CHAR)*4
RT_FORMAT_UNSIGNED_BYTE UCHAR
RT_FORMAT_UNSIGNED_BYTE2 sizeof(UCHAR)*2
RT_FORMAT_UNSIGNED_BYTE3 sizeof(UCHAR)*3
RT_FORMAT_UNSIGNED_BYTE4 sizeof(UCHAR)*4
RT_FORMAT_SHORT SHORT
RT_FORMAT_SHORT2 sizeof(SHORT)*2
RT_FORMAT_SHORT3 sizeof(SHORT)*3
RT_FORMAT_SHORT4 sizeof(SHORT)*4
RT_FORMAT_UNSIGNED_SHORT USHORT

RT_FORMAT_UNSIGNED_SHORT2 sizeof(USHORT)*2
RT_FORMAT_UNSIGNED_SHORT3 sizeof(USHORT)*3
RT_FORMAT_UNSIGNED_SHORT4 sizeof(USHORT)*4
RT_FORMAT_INT INT
RT_FORMAT_INT2 sizeof(INT)*2
RT_FORMAT_INT3 sizeof(INT)*3
RT_FORMAT_INT4 sizeof(INT)*4
RT_FORMAT_UNSIGNED_INT sizeof(UINT)
RT_FORMAT_UNSIGNED_INT2 sizeof(UINT)*2
RT_FORMAT_UNSIGNED_INT3 sizeof(UINT)*3
RT_FORMAT_UNSIGNED_INT4 sizeof(UINT)*4
RT_FORMAT_USER User Format
RT_FORMAT_BUFFER_ID Buffer Id
RT_FORMAT_PROGRAM_ID Program Id
RT_FORMAT_HALF half float
RT_FORMAT_HALF2 sizeof(half float)*2
RT_FORMAT_HALF3 sizeof(half float)*3
RT_FORMAT_HALF4 sizeof(half float)*4

3.4.2.11 enum RTgltarget

GL Target

Enumerator

RT_TARGET_GL_TEXTURE_2D GL texture 2D
RT_TARGET_GL_TEXTURE_RECTANGLE GL texture rectangle
RT_TARGET_GL_TEXTURE_3D GL texture 3D
RT_TARGET_GL_RENDER_BUFFER GL render buffer
RT_TARGET_GL_TEXTURE_1D GL texture 1D
RT_TARGET_GL_TEXTURE_1D_ARRAY GL array of 1D textures
RT_TARGET_GL_TEXTURE_2D_ARRAY GL array of 2D textures
RT_TARGET_GL_TEXTURE_CUBE_MAP GL cube map texture
RT_TARGET_GL_TEXTURE_CUBE_MAP_ARRAY GL array of cube maps

3.4.2.12 enum RTOBJECTTYPE

OptiX Object Types

Enumerator

RT_OBJECTTYPE_UNKNOWN Object Type Unknown
RT_OBJECTTYPE_GROUP Group Type
RT_OBJECTTYPE_GEOMETRY_GROUP Geometry Group Type
RT_OBJECTTYPE_TRANSFORM Transform Type
RT_OBJECTTYPE_SELECTOR Selector Type
RT_OBJECTTYPE_GEOMETRY_INSTANCE Geometry Instance Type
RT_OBJECTTYPE_BUFFER Buffer Type
RT_OBJECTTYPE_TEXTURE_SAMPLER Texture Sampler Type

RT_OBJECTTYPE_OBJECT Object Type
RT_OBJECTTYPE_MATRIX_FLOAT2x2 Matrix Float 2x2
RT_OBJECTTYPE_MATRIX_FLOAT2x3 Matrix Float 2x3
RT_OBJECTTYPE_MATRIX_FLOAT2x4 Matrix Float 2x4
RT_OBJECTTYPE_MATRIX_FLOAT3x2 Matrix Float 3x2
RT_OBJECTTYPE_MATRIX_FLOAT3x3 Matrix Float 3x3
RT_OBJECTTYPE_MATRIX_FLOAT3x4 Matrix Float 3x4
RT_OBJECTTYPE_MATRIX_FLOAT4x2 Matrix Float 4x2
RT_OBJECTTYPE_MATRIX_FLOAT4x3 Matrix Float 4x3
RT_OBJECTTYPE_MATRIX_FLOAT4x4 Matrix Float 4x4
RT_OBJECTTYPE_FLOAT Float Type
RT_OBJECTTYPE_FLOAT2 Float2 Type
RT_OBJECTTYPE_FLOAT3 Float3 Type
RT_OBJECTTYPE_FLOAT4 Float4 Type
RT_OBJECTTYPE_INT Integer Type
RT_OBJECTTYPE_INT2 Integer2 Type
RT_OBJECTTYPE_INT3 Integer3 Type
RT_OBJECTTYPE_INT4 Integer4 Type
RT_OBJECTTYPE_UNSIGNED_INT Unsigned Integer Type
RT_OBJECTTYPE_UNSIGNED_INT2 Unsigned Integer2 Type
RT_OBJECTTYPE_UNSIGNED_INT3 Unsigned Integer3 Type
RT_OBJECTTYPE_UNSIGNED_INT4 Unsigned Integer4 Type
RT_OBJECTTYPE_USER User Object Type
RT_OBJECTTYPE_PROGRAM Object Type Program - Added in OptiX 3.0

3.4.2.13 enum RTprogramidnull

Enumerator

RT_PROGRAM_ID_NULL sentinel for describing a non-existent program id

3.4.2.14 enum RTremotedeviceattribute

RemoteDevice attributes

Enumerator

RT_REMOTEDEVICE_ATTRIBUTE_CLUSTER_URL URL for the Cluster Manager
RT_REMOTEDEVICE_ATTRIBUTE_HEAD_NODE_URL URL for the Head Node
RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS Number of available configurations
RT_REMOTEDEVICE_ATTRIBUTE_STATUS Status
RT_REMOTEDEVICE_ATTRIBUTE_NUM_TOTAL_NODES Number of total nodes
RT_REMOTEDEVICE_ATTRIBUTE_NUM_FREE_NODES Number of free nodes
RT_REMOTEDEVICE_ATTRIBUTE_NUM_RESERVED_NODES Number of reserved nodes
RT_REMOTEDEVICE_ATTRIBUTE_NAME Name
RT_REMOTEDEVICE_ATTRIBUTE_NUM_GPUS Number of GPUs
RT_REMOTEDEVICE_ATTRIBUTE_GPU_TOTAL_MEMORY Total Memory (per GPU, in bytes)
RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS List of descriptions for the available configurations

3.4.2.15 enum RTremotedevicestatus

Enumerator

RT_REMOTEDEVICE_STATUS_READY RemoteDevice Status Ready
RT_REMOTEDEVICE_STATUS_CONNECTED RemoteDevice Status Connected
RT_REMOTEDEVICE_STATUS_RESERVED RemoteDevice Status Reserved
RT_REMOTEDEVICE_STATUS_DISCONNECTED RemoteDevice Status Disconnected

3.4.2.16 enum RTresult

Result

Enumerator

RT_SUCCESS Success
RT_TIMEOUT_CALLBACK Timeout callback
RT_ERROR_INVALID_CONTEXT Invalid Context
RT_ERROR_INVALID_VALUE Invalid Value
RT_ERROR_MEMORY_ALLOCATION_FAILED Timeout callback
RT_ERROR_TYPE_MISMATCH Type Mismatch
RT_ERROR_VARIABLE_NOT_FOUND Variable not found
RT_ERROR_VARIABLE_REDECLARED Variable redeclared
RT_ERROR_ILLEGAL_SYMBOL Illegal symbol
RT_ERROR_INVALID_SOURCE Invalid source
RT_ERROR_VERSION_MISMATCH Version mismatch
RT_ERROR_OBJECT_CREATION_FAILED Object creation failed
RT_ERROR_NO_DEVICE No device
RT_ERROR_INVALID_DEVICE Invalid device
RT_ERROR_INVALID_IMAGE Invalid image
RT_ERROR_FILE_NOT_FOUND File not found
RT_ERROR_ALREADY_MAPPED Already mapped
RT_ERROR_INVALID_DRIVER_VERSION Invalid driver version
RT_ERROR_CONTEXT_CREATION_FAILED Context creation failed
RT_ERROR_RESOURCE_NOT_REGISTERED Resource not registered
RT_ERROR_RESOURCE_ALREADY_REGISTERED Resource already registered
RT_ERROR_LAUNCH_FAILED Launch failed
RT_ERROR_NOT_SUPPORTED Not supported
RT_ERROR_CONNECTION_FAILED Connection failed
RT_ERROR_AUTHENTICATION_FAILED Authentication failed
RT_ERROR_CONNECTION_ALREADY_EXISTS Connection already exists
RT_ERROR_NETWORK_LOAD_FAILED Network component failed to load
RT_ERROR_NETWORK_INIT_FAILED Network initialization failed
RT_ERROR_CLUSTER_NOT_RUNNING No cluster is running
RT_ERROR_CLUSTER_ALREADY_RUNNING Cluster is already running
RT_ERROR_INSUFFICIENT_FREE_NODES Not enough free nodes
RT_ERROR_UNKNOWN Error unknown

3.4.2.17 enum RTtextureidnull

Enumerator

RT_TEXTURE_ID_NULL sentinel for describing a non-existent texture id

3.4.2.18 enum RTtextureindexmode

Texture index mode

Enumerator

RT_TEXTURE_INDEX_NORMALIZED_COORDINATES Texture Index normalized coordinates

RT_TEXTURE_INDEX_ARRAY_INDEX Texture Index Array

3.4.2.19 enum RTtexturereadmode

Texture read mode

Enumerator

RT_TEXTURE_READ_ELEMENT_TYPE Read element type

RT_TEXTURE_READ_NORMALIZED_FLOAT Read normalized float

RT_TEXTURE_READ_ELEMENT_TYPE_SRGB Read element type and apply sRGB to linear conversion during texture read for 8-bit integer buffer formats

RT_TEXTURE_READ_NORMALIZED_FLOAT_SRGB Read normalized float and apply sRGB to linear conversion during texture read for 8-bit integer buffer formats

3.4.2.20 enum RTwrapmode

Wrap mode

Enumerator

RT_WRAP_REPEAT Wrap repeat

RT_WRAP_CLAMP_TO_EDGE Clamp to edge

RT_WRAP_MIRROR Mirror

RT_WRAP_CLAMP_TO_BORDER Clamp to border

3.5 optix_defines.h File Reference

3.5.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Definitions

Classes

- struct [rti_internal_typeinfo::rti_typeinfo](#)
- struct [rti_internal_typeinfo::rti_typeenum< T >](#)

Macros

- #define **OPTIX_ASM_PTR** "r"
- #define **OPTIX_ASM_SIZE_T** "r"
- #define **OPTIX_ASM_PTR_SIZE_STR** "32"
- #define **OPTIX_BITNESS_SUFFIX** ""

Typedefs

- typedef size_t **optix::optix_size_t**

Enumerations

- enum **RTtransformkind** {
RT_WORLD_TO_OBJECT = 0xf00,
RT_OBJECT_TO_WORLD }
- enum **RTtransformflags** { **RT_INTERNAL_INVERSE_TRANSPOSE** = 0x1000 }
- enum **rtiTypeKind** { **_OPTIX_VARIABLE** = 0x796152 }
- enum **rtiTypeEnum** {
_OPTIX_TYPE_ENUM_UNKNOWN = 0x1337,
_OPTIX_TYPE_ENUM_PROGRAM_ID,
_OPTIX_TYPE_ENUM_PROGRAM_AS_ID }
- enum **rtiTexLookupKind** {
TEX_LOOKUP_1D = 1,
TEX_LOOKUP_2D = 2,
TEX_LOOKUP_3D = 3,
TEX_LOOKUP_A1 = 4,
TEX_LOOKUP_A2 = 5,
TEX_LOOKUP_CUBE = 6,
TEX_LOOKUP_ACUBE = 7 }

3.5.2 Enumeration Type Documentation

3.5.2.1 enum RTtransformflags

Transform flags

Enumerator

RT_INTERNAL_INVERSE_TRANSPOSE Inverse transpose flag

3.5.2.2 enum RTtransformkind

Transform type

Enumerator

RT_WORLD_TO_OBJECT World to Object transformation

RT_OBJECT_TO_WORLD Object to World transformation

3.6 optix_device.h File Reference

3.6.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Host/Device side

Classes

- struct [optix::VectorTypes< T, Dim >](#)
- struct [optix::VectorTypes< int, 1 >](#)
- struct [optix::VectorTypes< int, 2 >](#)
- struct [optix::VectorTypes< int, 3 >](#)
- struct [optix::VectorTypes< int, 4 >](#)
- struct [optix::VectorTypes< unsigned int, 1 >](#)
- struct [optix::VectorTypes< unsigned int, 2 >](#)
- struct [optix::VectorTypes< unsigned int, 3 >](#)
- struct [optix::VectorTypes< unsigned int, 4 >](#)
- struct [optix::VectorTypes< float, 1 >](#)
- struct [optix::VectorTypes< float, 2 >](#)
- struct [optix::VectorTypes< float, 3 >](#)
- struct [optix::VectorTypes< float, 4 >](#)
- struct [rtObject](#)
- struct [rtCallableProgramSizeofWrapper< T >](#)
- struct [rtCallableProgramSizeofWrapper< void >](#)
- struct [optix::bufferId< T, Dim >](#)
- struct [optix::buffer< T, Dim >](#)
- struct [optix::buffer< T, Dim >::type< T2 >](#)
- struct [optix::bufferId< T, Dim >](#)
- class [rti_internal_callableprogram::CPArgVoid](#)
- struct [rti_internal_callableprogram::is_CPArgVoid< T1 >](#)
- struct [rti_internal_callableprogram::is_CPArgVoid< CPArgVoid >](#)
- struct [rti_internal_callableprogram::check_is_CPArgVoid< Condition, Dummy >](#)
- struct [rti_internal_callableprogram::check_is_CPArgVoid< false, IntentionalError >](#)
- class [rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >](#)
- class [optix::callableProgramId< T >](#)
- class [optix::boundCallableProgramId< T >](#)
- struct [rti_internal_typeinfo::rti_typeenum< optix::callableProgramId< T > >](#)
- struct [rti_internal_typeinfo::rti_typeenum< optix::boundCallableProgramId< T > >](#)

Macros

- [#define rtDeclareVariable](#)(type, name, semantic, annotation)
- [#define rtDeclareAnnotation](#)(variable, annotation)
- [#define rtCallableProgram](#)(return_type, function_name, parameter_list)
- [#define rtBuffer](#) __device__ [optix::buffer](#)
- [#define rtBufferId](#) [optix::bufferId](#)
- [#define rtTextureSampler](#) texture
- [#define _OPTIX_TEX_FUNC_DECLARE_](#)(FUNC, SIGNATURE, PARAMS)
- [#define RT_PROGRAM](#) __global__
- [#define RT_CALLABLE_PROGRAM](#) __device__ __noinline__
- [#define RT_INTERNAL_CALLABLE_PROGRAM_DEFS](#)(...)
- [#define RT_INTERNAL_BOUND_CALLABLE_PROGRAM_DEFS](#)(...)
- [#define rtCallableProgramId](#) [optix::callableProgramId](#)
- [#define rtCallableProgramX](#) [optix::boundCallableProgramId](#)

Typedefs

- typedef int **optix::rtTextureId**

Functions

- `__device__ int4 optix::float4AsInt4 (float4 f4)`
- `__device__ uint4 optix::float4AsUInt4 (float4 f4)`
- `template<class T >`
`static __device__ void rtTrace (rtObject topNode, optix::Ray ray, T &prd)`
- `static __device__ bool rtPotentialIntersection (float tmin)`
- `static __device__ bool rtReportIntersection (unsigned int material)`
- `static __device__ void rtIgnoreIntersection ()`
- `static __device__ void rtTerminateRay ()`
- `static __device__ void rtIntersectChild (unsigned int index)`
- `static __device__ float3 rtTransformPoint (RTtransformkind kind, const float3 &p)`
- `static __device__ float3 rtTransformVector (RTtransformkind kind, const float3 &v)`
- `static __device__ float3 rtTransformNormal (RTtransformkind kind, const float3 &n)`
- `static __device__ void rtGetTransform (RTtransformkind kind, float matrix[16])`
- `static __device__ void rtThrow (unsigned int code)`
- `static __device__ unsigned int rtGetExceptionCode ()`
- `static __device__ void rtPrintExceptionDetails ()`
- `static __device__ void rtPrintf (const char *fmt)`
- `template<typename T1 >`
`static __device__ void rtPrintf (const char *fmt, T1 arg1)`
- `template<typename T1, typename T2 >`
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2)`
- `template<typename T1, typename T2, typename T3 >`
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3)`
- `template<typename T1, typename T2, typename T3, typename T4 >`
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4)`
- `template<typename T1, typename T2, typename T3, typename T4, typename T5 >`
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5)`
- `template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6 >`
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6)`
- `template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7 >`
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7)`
- `template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8 >`
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8)`
- `template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9 >`
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9)`
- `template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10 >`
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10)`
- `template<typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8, typename T9, typename T10, typename T11 >`
`static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10, T11 arg11)`

- `template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 ,
typename T9 , typename T10 , typename T11 , typename T12 >
static __device__ void rtPrintf (const char *fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7
arg7, T8 arg8, T9 arg9, T10 arg10, T11 arg11, T12 arg12)`

- `rtTextureId optix::id`
- `rtTextureId float optix::x`
- `* optix::retVal = tmp`
- `rtTextureId float float optix::y`
- `rtTextureId float float float optix::z`
- `rtTextureId float float int optix::comp`
- `rtTextureId float float optix::dPdx`
- `rtTextureId float float float optix::dPdy`
- `rtTextureId float int optix::layer`
- `rtTextureId float float optix::level`
- `__device__ uint3 optix::rtTexSize (rtTextureId id)`
- `template<typename T >
__device__ T optix::rtTex1D (rtTextureId id, float x)`
- `template<>
__device__ float4 optix::rtTex1D (rtTextureId id, float x)`
- `template<>
__device__ int4 optix::rtTex1D (rtTextureId id, float x)`
- `template<>
__device__ uint4 optix::rtTex1D (rtTextureId id, float x)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex1D,(rtTextureId id, float x),(id, x)) template< typename T >
inline __device__ void rtTex1D(T *retVal`
- `template<typename T >
__device__ T optix::rtTex1DFetch (rtTextureId id, int x)`
- `template<>
__device__ float4 optix::rtTex1DFetch (rtTextureId id, int x)`
- `template<>
__device__ int4 optix::rtTex1DFetch (rtTextureId id, int x)`
- `template<>
__device__ uint4 optix::rtTex1DFetch (rtTextureId id, int x)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex1DFetch,(rtTextureId id, int x),(id, x)) template< typename T
> inline __device__ void rtTex1DFetch(T *retVal`
- `template<typename T >
__device__ T optix::rtTex2D (rtTextureId id, float x, float y)`
- `template<>
__device__ float4 optix::rtTex2D (rtTextureId id, float x, float y)`
- `template<>
__device__ int4 optix::rtTex2D (rtTextureId id, float x, float y)`
- `template<>
__device__ uint4 optix::rtTex2D (rtTextureId id, float x, float y)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex2D,(rtTextureId id, float x, float y),(id, x, y)) template< type-
name T > inline __device__ void rtTex2D(T *retVal`
- `template<typename T >
__device__ T optix::rtTex2DFetch (rtTextureId id, int x, int y)`
- `template<>
__device__ float4 optix::rtTex2DFetch (rtTextureId id, int x, int y)`
- `template<>
__device__ int4 optix::rtTex2DFetch (rtTextureId id, int x, int y)`
- `template<>
__device__ uint4 optix::rtTex2DFetch (rtTextureId id, int x, int y)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex2DFetch,(rtTextureId id, int x, int y),(id, x, y)) template< type-
name T > inline __device__ void rtTex2DFetch(T *retVal`

- `template<typename T >`
`__device__ T optix::rtTex3D (rtTextureId id, float x, float y, float z)`
- `template<>`
`__device__ float4 optix::rtTex3D (rtTextureId id, float x, float y, float z)`
- `template<>`
`__device__ int4 optix::rtTex3D (rtTextureId id, float x, float y, float z)`
- `template<>`
`__device__ uint4 optix::rtTex3D (rtTextureId id, float x, float y, float z)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex3D,(rtTextureId id, float x, float y, float z),(id, x, y, z))`
`template< typename T > inline __device__ void rtTex3D(T *retVal`
- `template<typename T >`
`__device__ T optix::rtTex3DFetch (rtTextureId id, int x, int y, int z)`
- `template<>`
`__device__ float4 optix::rtTex3DFetch (rtTextureId id, int x, int y, int z)`
- `template<>`
`__device__ int4 optix::rtTex3DFetch (rtTextureId id, int x, int y, int z)`
- `template<>`
`__device__ uint4 optix::rtTex3DFetch (rtTextureId id, int x, int y, int z)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex3DFetch,(rtTextureId id, int x, int y, int z),(id, x, y, z))`
`template< typename T > inline __device__ void rtTex3DFetch(T *retVal`
- `template<typename T >`
`__device__ T optix::rtTex2DGather (rtTextureId id, float x, float y, int comp=0)`
- `template<>`
`__device__ float4 optix::rtTex2DGather (rtTextureId id, float x, float y, int comp)`
- `template<>`
`__device__ int4 optix::rtTex2DGather (rtTextureId id, float x, float y, int comp)`
- `template<>`
`__device__ uint4 optix::rtTex2DGather (rtTextureId id, float x, float y, int comp)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex2DGather,(rtTextureId id, float x, float y, int comp),(id, x, y, comp))`
`template< typename T > inline __device__ void rtTex2DGather(T *retVal`
- `template<>`
`__device__ float4 optix::rtTex1DGrad (rtTextureId id, float x, float dPdx, float dPdy)`
- `template<>`
`__device__ int4 optix::rtTex1DGrad (rtTextureId id, float x, float dPdx, float dPdy)`
- `template<>`
`__device__ uint4 optix::rtTex1DGrad (rtTextureId id, float x, float dPdx, float dPdy)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex1DGrad,(rtTextureId id, float x, float dPdx, float dPdy),(id, x, dPdx, dPdy))`
`template< typename T > inline __device__ void rtTex1DGrad(T *retVal`
- `template<typename T >`
`__device__ T optix::rtTex2DGrad (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)`
- `template<>`
`__device__ float4 optix::rtTex2DGrad (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)`
- `template<>`
`__device__ int4 optix::rtTex2DGrad (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)`
- `template<>`
`__device__ uint4 optix::rtTex2DGrad (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex2DGrad,(rtTextureId id, float x, float y, float2 dPdx, float2 dPdy),(id, x, y, dPdx, dPdy))`
`template< typename T > inline __device__ void rtTex2DGrad(T *retVal`
- `template<typename T >`
`__device__ T optix::rtTex3DGrad (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)`
- `template<>`
`__device__ float4 optix::rtTex3DGrad (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)`
- `template<>`
`__device__ int4 optix::rtTex3DGrad (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)`
- `template<>`
`__device__ uint4 optix::rtTex3DGrad (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)`

- **optix::OPTIX_TEX_FUNC_DECLARE_** (rtTex3DGrad,(rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy),(id, x, y, z, dPdx, dPdy)) template< typename T > inline __device__ void rtTex3DGrad(T *retVal
- template<typename T >
__device__ T **optix::rtTex1DLayeredGrad** (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- template<>
__device__ float4 **optix::rtTex1DLayeredGrad** (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- template<>
__device__ int4 **optix::rtTex1DLayeredGrad** (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- template<>
__device__ uint4 **optix::rtTex1DLayeredGrad** (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- **optix::OPTIX_TEX_FUNC_DECLARE_** (rtTex1DLayeredGrad,(rtTextureId id, float x, int layer, float dPdx, float dPdy),(id, x, layer, dPdx, dPdy)) template< typename T > inline __device__ void rtTex1DLayeredGrad(T *retVal
- template<typename T >
__device__ T **optix::rtTex2DLayeredGrad** (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- template<>
__device__ float4 **optix::rtTex2DLayeredGrad** (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- template<>
__device__ int4 **optix::rtTex2DLayeredGrad** (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- template<>
__device__ uint4 **optix::rtTex2DLayeredGrad** (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- **optix::OPTIX_TEX_FUNC_DECLARE_** (rtTex2DLayeredGrad,(rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy),(id, x, y, layer, dPdx, dPdy)) template< typename T > inline __device__ void rtTex2DLayeredGrad(T *retVal
- template<typename T >
__device__ T **optix::rtTex1DLod** (rtTextureId id, float x, float level)
- template<>
__device__ float4 **optix::rtTex1DLod** (rtTextureId id, float x, float level)
- template<>
__device__ int4 **optix::rtTex1DLod** (rtTextureId id, float x, float level)
- template<>
__device__ uint4 **optix::rtTex1DLod** (rtTextureId id, float x, float level)
- **optix::OPTIX_TEX_FUNC_DECLARE_** (rtTex1DLod,(rtTextureId id, float x, float level),(id, x, level)) template< typename T > inline __device__ void rtTex1DLod(T *retVal
- template<typename T >
__device__ T **optix::rtTex2DLod** (rtTextureId id, float x, float y, float level)
- template<>
__device__ float4 **optix::rtTex2DLod** (rtTextureId id, float x, float y, float level)
- template<>
__device__ int4 **optix::rtTex2DLod** (rtTextureId id, float x, float y, float level)
- template<>
__device__ uint4 **optix::rtTex2DLod** (rtTextureId id, float x, float y, float level)
- **optix::OPTIX_TEX_FUNC_DECLARE_** (rtTex2DLod,(rtTextureId id, float x, float y, float level),(id, x, y, level)) template< typename T > inline __device__ void rtTex2DLod(T *retVal
- template<typename T >
__device__ T **optix::rtTex3DLod** (rtTextureId id, float x, float y, float z, float level)
- template<>
__device__ float4 **optix::rtTex3DLod** (rtTextureId id, float x, float y, float z, float level)
- template<>
__device__ int4 **optix::rtTex3DLod** (rtTextureId id, float x, float y, float z, float level)
- template<>
__device__ uint4 **optix::rtTex3DLod** (rtTextureId id, float x, float y, float z, float level)
- **optix::OPTIX_TEX_FUNC_DECLARE_** (rtTex3DLod,(rtTextureId id, float x, float y, float z, float level),(id, x, y, z, level)) template< typename T > inline __device__ void rtTex3DLod(T *retVal

- `template<typename T >`
`__device__ T optix::rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)`
- `template<>`
`__device__ float4 optix::rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)`
- `template<>`
`__device__ int4 optix::rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)`
- `template<>`
`__device__ uint4 optix::rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex1DLayeredLod,(rtTextureId id, float x, int layer, float level),(id, x, layer, level))` `template< typename T > inline __device__ void rtTex1DLayeredLod(T *retVal`
- `template<typename T >`
`__device__ T optix::rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- `template<>`
`__device__ float4 optix::rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- `template<>`
`__device__ int4 optix::rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- `template<>`
`__device__ uint4 optix::rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex2DLayeredLod,(rtTextureId id, float x, float y, int layer, float level),(id, x, y, layer, level))` `template< typename T > inline __device__ void rtTex2DLayeredLod(T *retVal`
- `template<typename T >`
`__device__ T optix::rtTex1DLayered (rtTextureId id, float x, int layer)`
- `template<>`
`__device__ float4 optix::rtTex1DLayered (rtTextureId id, float x, int layer)`
- `template<>`
`__device__ int4 optix::rtTex1DLayered (rtTextureId id, float x, int layer)`
- `template<>`
`__device__ uint4 optix::rtTex1DLayered (rtTextureId id, float x, int layer)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex1DLayered,(rtTextureId id, float x, int layer),(id, x, layer))` `template< typename T > inline __device__ void rtTex1DLayered(T *retVal`
- `template<typename T >`
`__device__ T optix::rtTex2DLayered (rtTextureId id, float x, float y, int layer)`
- `template<>`
`__device__ float4 optix::rtTex2DLayered (rtTextureId id, float x, float y, int layer)`
- `template<>`
`__device__ int4 optix::rtTex2DLayered (rtTextureId id, float x, float y, int layer)`
- `template<>`
`__device__ uint4 optix::rtTex2DLayered (rtTextureId id, float x, float y, int layer)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTex2DLayered,(rtTextureId id, float x, float y, int layer),(id, x, y, layer))` `template< typename T > inline __device__ void rtTex2DLayered(T *retVal`
- `template<typename T >`
`__device__ T optix::rtTexCubemap (rtTextureId id, float x, float y, float z)`
- `template<>`
`__device__ float4 optix::rtTexCubemap (rtTextureId id, float x, float y, float z)`
- `template<>`
`__device__ int4 optix::rtTexCubemap (rtTextureId id, float x, float y, float z)`
- `template<>`
`__device__ uint4 optix::rtTexCubemap (rtTextureId id, float x, float y, float z)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTexCubemap,(rtTextureId id, float x, float y, float z),(id, x, y, z))` `template< typename T > inline __device__ void rtTexCubemap(T *retVal`
- `template<typename T >`
`__device__ T optix::rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`
- `template<>`
`__device__ float4 optix::rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`
- `template<>`
`__device__ int4 optix::rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`

- `template<>`
`__device__ uint4 optix::rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTexCubemapLayered,(rtTextureId id, float x, float y, float z, int layer),(id, x, y, z, layer))` `template< typename T > inline __device__ void rtTexCubemapLayered(T *retVal`
- `template<typename T >`
`__device__ T optix::rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`
`__device__ float4 optix::rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`
`__device__ int4 optix::rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`
`__device__ uint4 optix::rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTexCubemapLod,(rtTextureId id, float x, float y, float z, float level),(id, x, y, z, level))` `template< typename T > inline __device__ void rtTexCubemapLod(T *retVal`
- `template<typename T >`
`__device__ T optix::rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- `template<>`
`__device__ float4 optix::rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- `template<>`
`__device__ int4 optix::rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- `template<>`
`__device__ uint4 optix::rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- `optix::OPTIX_TEX_FUNC_DECLARE_ (rtTexCubemapLayeredLod,(rtTextureId id, float x, float y, float z, int layer, float level),(id, x, y, z, layer, level))` `template< typename T > inline __device__ void rtTexCubemapLayeredLod(T *retVal`

3.6.2 Macro Definition Documentation

3.6.2.1 #define RT_INTERNAL_BOUND_CALLABLE_PROGRAM_DEFS(...)

Value:

```
public rti_internal_callableprogram::callableProgramIdBase<__VA_ARGS__>
{
}
```

3.6.2.2 #define RT_INTERNAL_CALLABLE_PROGRAM_DEFS(...)

Value:

```
public rti_internal_callableprogram::callableProgramIdBase<__VA_ARGS__>
{
public:
    /* Default constructor */
    __device__ __forceinline__ callableProgramId() {}
    /* Constructor that initializes the id with null.*/
    __device__ __forceinline__ callableProgramId(RTprogramidnull nullid)
    : rti_internal_callableprogram::callableProgramIdBase<__VA_ARGS__>
      (nullid) {} \
    /* Constructor that initializes the id.*/
    __device__ __forceinline__ explicit callableProgramId(int id)
    : rti_internal_callableprogram::callableProgramIdBase<__VA_ARGS__>
      (id) {} \
    /* assignment that initializes the id with null. */
    __device__ __forceinline__ callableProgramId& operator= (RTprogramidnull nullid) \
    { this->m_id = nullid; return *this; } \
    /* Return the id */
    __device__ __forceinline__ int getId() const { return this->m_id; } \
    /* Return whether the id is valid */
    __device__ __forceinline__ operator bool() const \
    { return this->m_id != RT_PROGRAM_ID_NULL; } \
}
```

3.6.2.3 `#define RT_PROGRAM __global__`

Define an OptiX program.

Description

`RT_PROGRAM` defines a program **program_name** with the specified arguments and return value. This function can be bound to a specific program object using `rtProgramCreateFromPTXString` or `rtProgramCreateFromPTXFile`, which will subsequently get bound to different programmable binding points.

All programs should have a "void" return type. Bounding box programs will have an argument for the primitive index and the bounding box reference return value (type `nvrt::AAbb&`). Intersection programs will have a single `int primitiveIndex` argument. All other programs take zero arguments.

History

`RT_PROGRAM` was introduced in OptiX 1.0.

See also `RT_PROGRAM` `rtProgramCreateFromPTXFile` `rtProgramCreateFromPTXString`

3.6.2.4 `#define rtBuffer __device__ optix::buffer`

Declare a reference to a buffer object.

Description

```
rtBuffer<Type, Dim> name;
```

`rtBuffer` declares a buffer of type *Type* and dimensionality *Dim*. *Dim* must be between 1 and 4 inclusive and defaults to 1 if not specified. The resulting object provides access to buffer data through the `[]` indexing operator, where the index is either unsigned int, uint2, uint3, or uint4 for 1, 2, 3 or 4-dimensional buffers (respectively). This operator can be used to read from or write to the resulting buffer at the specified index.

The named buffer obeys the runtime name lookup semantics as described in `rtDeclareVariable`. A compile error will result if the named buffer is not bound to a buffer object, or is bound to a buffer object of the incorrect type or dimension. The behavior of writing to a read-only buffer is undefined. Reading from a write-only buffer is well defined only if a value has been written previously by the same thread.

This declaration must appear at the file scope (not within a function), and will be visible to all `RT_PROGRAM` instances within the same compilation unit.

An annotation may be associated with the buffer variable by using the `rtDeclareAnnotation` macro.

History

`rtBuffer` was introduced in OptiX 1.0.

See also `rtDeclareAnnotation`, `rtDeclareVariable`, `rtBufferCreate`, `rtTextureSampler`, `rtVariableSetObject` `rtBufferId`

3.6.2.5 `#define rtBufferId optix::bufferId`

A class that wraps buffer access functionality when using a buffer id.

Description

The `rtBufferId` provides an interface similar to `rtBuffer` when using a buffer id obtained through `rtBufferGetId`. Unlike `rtBuffer`, this class can be passed to functions or stored in other data structures such as the ray payload. It should be noted, however, doing so can limit the extent that OptiX can optimize the generated code.

There is also a version of `rtBufferId` that can be used by the host code, so that types can exist in both host and device code. See the documentation for `rtBufferId` found in the optix C++ API header.

History

`rtBufferId` was introduced in OptiX 3.5.

See also

`rtBuffer` `rtBufferGetId`

3.6.2.6 #define rtCallableProgram(return_type, function_name, parameter_list)

Value:

```
namespace rti_internal_typeinfo {
    __device__ ::rti_internal_typeinfo::rti_typeinfo function_name = {
        ::rti_internal_typeinfo::_OPTIX_VARIABLE, rtCallableProgramSizeofWrapper<return_type>::value }
    ; \
}
namespace rti_internal_typename {
    __device__ char function_name[] = #return_type;
}
namespace rti_internal_semantic {
    __device__ char function_name[] = ""; /* used to be rt_call, but not needed anymore */ \
}
namespace rti_internal_annotation {
    __device__ char function_name[] = #parameter_list;
}
__noinline__ __device__ return_type function_name parameter_list { typedef return_type localtype; return
    localtype(); }
```

Callable Program Declaration.

Description

[rtCallableProgram](#) declares callable program *name*, which will appear to be a callable function with the specified return type and list of arguments. This callable program must be matched against a variable declared on the API object using [rtVariableSetObject](#).

Unless compatibility with SM_10 is needed, new code should #define RT_USE_TEMPLATED_RTCALLABLEPROGRAM and rely on the new templated version of [rtCallableProgram](#).

Example(s):

```
rtCallableProgram(float3, modColor, (float3, float));
```

Parameters

| | | |
|----|-----------------------|--|
| in | <i>return_type</i> | Return type of the callable program |
| in | <i>function_name</i> | Name of the callable program |
| in | <i>parameter_list</i> | Parameter_List of the callable program |

History

[rtCallableProgram](#) was introduced in OptiX 3.0.

See also [rtDeclareVariable](#) [rtCallableProgramId](#) [rtCallableProgramX](#)

3.6.2.7 #define rtCallableProgramId optix::callableProgramId

Callable Program ID Declaration.

Description

[rtCallableProgramId](#) declares callable program *name*, which will appear to be a callable function with the specified return type and list of arguments. This callable program must be matched against a variable declared on the API object of type int.

Example(s):

```
rtDeclareVariable(rtCallableProgramId<float3(float3, float)>, modColor)
;
rtBuffer<rtCallableProgramId<float3(float3, float)>, 1> modColors;
```

History

[rtCallableProgramId](#) was introduced in OptiX 3.6.

See also [rtCallableProgram](#) [rtCallableProgramX](#) [rtDeclareVariable](#)

3.6.2.8 #define rtCallableProgramX optix::boundCallableProgramId

Callable Program X Declaration.

Description

[rtCallableProgramX](#) declares callable program *name*, which will appear to be a callable function with the specified return type and list of arguments. This callable program must be matched against a variable declared on the API object using [rtVariableSetObject](#).

Unless compatibility with SM_10 is needed, new code should #define RT_USE_TEMPLATED_RTCALLABLEPROGRAM and rely on the new templated version of [rtCallableProgram](#) instead of directly using [rtCallableProgramX](#).

Example(s):

```
rtDeclareVariable(rtCallableProgramX<float3(float3, float)>, modColor);
// With RT_USE_TEMPLATED_RTCALLABLEPROGRAM defined
rtDeclareVariable(rtCallableProgram<float3(float3, float)>, modColor);
```

History

[rtCallableProgramX](#) was introduced in OptiX 3.6.

See also [rtCallableProgram](#) [rtCallableProgramId](#) [rtDeclareVariable](#)

3.6.2.9 #define rtDeclareAnnotation(variable, annotation)

Value:

```
namespace rti_internal_annotation { \
    __device__ char variable[] = #annotation; \
}
```

Annotation declaration.

Description

[rtDeclareAnnotation](#) sets the annotation *annotation* of the given variable *name*. Typically annotations are declared using an argument to [rtDeclareVariable](#), but variables of type [rtBuffer](#) and [rtTextureSampler](#) are declared using templates, so separate annotation attachment is required.

OptiX does not attempt to interpret the annotation in any way. It is considered metadata for the application to query and interpret in its own way.

Valid annotations

The macro [rtDeclareAnnotation](#) uses the C pre-processor's "stringification" feature to turn the literal text of the annotation argument into a string constant. The pre-processor will backslash-escape quotes and backslashes within the text of the annotation. Leading and trailing whitespace will be ignored, and sequences of whitespace in the middle of the text is converted to a single space character in the result. The only restriction the C-PP places on the text is that it may not contain a comma character unless it is either quoted or contained within parens: "," or (.).

Example(s):

```
rtDeclareAnnotation( tex, this is a test );
annotation = "this is a test"

rtDeclareAnnotation( tex, "this is a test" );
annotation = "\"this is a test\""

rtDeclareAnnotation( tex, float3 a = {1, 2, 3} );
--> Compile Error, no unquoted commas may be present in the annotation

rtDeclareAnnotation( tex, "float3 a = {1, 2, 3}" );
annotation = "\"float3 a = {1, 2, 3}\""

rtDeclareAnnotation( tex, string UIWidget = "slider";
                    float UIMin = 0.0;
                    float UIMax = 1.0; );
annotation = "string UIWidget = \"slider\"; float UIMin = 0.0; float UIMax = 1.0;"
```

Parameters

| | | |
|----|-------------------|----------------------|
| in | <i>variable</i> | Variable to annotate |
| in | <i>annotation</i> | Annotation metadata |

History

[rtDeclareAnnotation](#) was introduced in OptiX 1.0.

See also [rtDeclareVariable](#), [rtVariableGetAnnotation](#)

3.6.2.10 #define rtDeclareVariable(type, name, semantic, annotation)

Value:

```
namespace rti_internal_typeinfo { \
    __device__ ::rti_internal_typeinfo::rti_typeinfo name = { ::rti_internal_typeinfo::_OPTIX_VARIABLE,
        sizeof(type) }; \
} \
namespace rti_internal_typename { \
    __device__ char name[] = #type; \
} \
namespace rti_internal_typeenum { \
    __device__ int name =
        ::rti_internal_typeinfo::rti_typeenum<type>::_m_typeenum
    ; \
} \
namespace rti_internal_semantic { \
    __device__ char name[] = #semantic; \
} \
namespace rti_internal_annotation { \
    __device__ char name[] = #annotation; \
} \
__device__ type name
```

Variable declaration.

Description

[rtDeclareVariable](#) declares variable *name* of the specified *type*. By default, the variable name will be matched against a variable declared on the API object using the lookup hierarchy for the current program. Using the semantic-Name, this variable can be bound to internal state, to the payload associated with a ray, or to attributes that are communicated between intersection and material programs. An additional optional annotation can be used to associate application-specific metadata with the variable as well.

type may be a primitive type or a user-defined struct (See [rtVariableSetUserData](#)). Except for the ray payload and attributes, the declared variable will be read-only. The variable will be visible to all of the cuda functions defined in the current file. The binding of variables to values on API objects is allowed to vary from one instance to another.

Valid semanticNames

- **rtLaunchIndex** - The launch invocation index. Type must be one of *unsigned int*, *uint2*, *uint3*, *int*, *int2*, *int3* and is read-only.
- **rtLaunchDim** - The size of each dimension of the launch. The values range from 1 to the launch size in that dimension. Type must be one of *unsigned int*, *uint2*, *uint3*, *int*, *int2*, *int3* and is read-only.
- **rtCurrentRay** - The currently active ray, valid only when a call to [rtTrace](#) is active. Type must be *optix::Ray* and is read-only.
- **rtIntersectionDistance** - The current closest hit distance, valid only when a call to [rtTrace](#) is active. Type must be *float* and is read-only.
- **rtRayPayload** - The struct passed into the most recent [rtTrace](#) call and is read-write.
- **attribute name** - A named attribute passed from the intersection program to a closest-hit or any-hit program. The types must match in both sets of programs. This variable is read-only in the closest-hit or any-hit program and is written in the intersection program.

Parameters

| | | |
|----|-------------------|------------------------------|
| in | <i>type</i> | Type of the variable |
| in | <i>name</i> | Name of the variable |
| in | <i>semantic</i> | Semantic name |
| in | <i>annotation</i> | Annotation for this variable |

History

- [rtDeclareVariable](#) was introduced in OptiX 1.0.
- [rtLaunchDim](#) was introduced in OptiX 2.0.

See also [rtDeclareAnnotation](#), [rtVariableGetAnnotation](#), [rtContextDeclareVariable](#), [rtProgramDeclareVariable](#), [rtSelectorDeclareVariable](#), [rtGeometryInstanceDeclareVariable](#), [rtGeometryDeclareVariable](#), [rtMaterialDeclareVariable](#)

3.6.2.11 #define rtTextureSampler texture

Declares a reference to a texture sampler object.

Description

[rtTextureSampler](#) declares a texture of type *Type* and dimensionality *Dim*. *Dim* must be between 1 and 3 inclusive and defaults to 1 if not specified. The resulting object provides access to texture data through the `tex1D`, `tex2D` and `tex3D` functions. These functions can be used only to read the data.

Texture filtering and wrapping modes, specified in *ReadMode* will be dependent on the state of the texture sampler object created with [rtTextureSamplerCreate](#).

An annotation may be associated with the texture sampler variable by using the [rtDeclareAnnotation](#) macro.

History

[rtTextureSampler](#) was introduced in OptiX 1.0.

See also [rtDeclareAnnotation](#), [rtTextureSamplerCreate](#)

3.6.3 Function Documentation

3.6.3.1 static __device__ unsigned int rtGetExceptionCode () [inline],[static]

Retrieves the type of a caught exception.

Description

[rtGetExceptionCode](#) can be called from an exception program to query which type of exception was caught. The returned code is equivalent to one of the [RTException](#) constants passed to [rtContextSetExceptionEnabled](#), [RT_EXCEPTION_ALL](#) excluded. For user-defined exceptions, the code is equivalent to the argument passed to [rtThrow](#).

Return values

| | |
|-----------------|-----------------------------|
| <i>unsigned</i> | int Returned exception code |
|-----------------|-----------------------------|

History

[rtGetExceptionCode](#) was introduced in OptiX 1.1.

See also [rtContextSetExceptionEnabled](#), [rtContextGetExceptionEnabled](#), [rtContextSetExceptionProgram](#), [rtContextGetExceptionProgram](#), [rtThrow](#), [rtPrintExceptionDetails](#)

3.6.3.2 static __device__ void rtGetTransform (RTtransformkind kind, float matrix[16]) [inline],[static]

Get requested transform.

Description

[rtGetTransform](#) returns the requested transform in the return parameter *matrix*. The type of transform to be retrieved is specified with the *kind* parameter. *kind* is an enumerated value that can be either [RT_OBJECT_TO_WORLD](#) or [RT_WORLD_TO_OBJECT](#) and must be a constant literal. During traversal, intersection and any-hit programs, the current ray will be located in object space. During ray generation, closest-hit and miss programs, the current ray will be located in world space.

There may be significant performance overhead associated with a call to [rtGetTransform](#) compared to a call to [rtTransformPoint](#), [rtTransformVector](#), or [rtTransformNormal](#).

Parameters

| | | |
|-----|---------------|--|
| in | <i>kind</i> | The type of transform to retrieve |
| out | <i>matrix</i> | Return parameter for the requested transform |

Return values

| | |
|-------------|-------------------|
| <i>void</i> | void return value |
|-------------|-------------------|

History

[rtGetTransform](#) was introduced in OptiX 1.0.

See also [rtTransformCreate](#), [rtTransformPoint](#), [rtTransformVector](#), [rtTransformNormal](#)

3.6.3.3 static __device__ void rtIgnoreIntersection () [inline],[static]

Cancels the potential intersection with current ray.

Description

[rtIgnoreIntersection](#) causes the current potential intersection to be ignored. This intersection will not become the new closest hit associated with the ray. This function does not return, so values affecting the per-ray data should be applied before calling [rtIgnoreIntersection](#). [rtIgnoreIntersection](#) is valid only within an any-hit program.

[rtIgnoreIntersection](#) can be used to implement alpha-mapped transparency by ignoring intersections that hit the geometry but are labeled as transparent in a texture. Since any-hit programs are called frequently during intersection, care should be taken to make them as efficient as possible.

Return values

| | |
|-------------|-------------------|
| <i>void</i> | void return value |
|-------------|-------------------|

History

[rtIgnoreIntersection](#) was introduced in OptiX 1.0.

See also [rtTerminateRay](#), [rtPotentialIntersection](#)

3.6.3.4 static __device__ void rtIntersectChild (unsigned int *index*) [inline],[static]

Visit child of selector.

Description

[rtIntersectChild](#) will perform intersection on the specified child for the current active ray. This is used in a selector visit program to traverse one of the selector's children. The *index* specifies which of the children to be visited. As the child is traversed, intersection programs will be called and any-hit programs will be called for positive intersections. When this process is complete, [rtIntersectChild](#) will return unless one of the any-hit programs calls [rtTerminateRay](#), in which case this function will never return. Multiple children can be visited during a single selector visit call by calling this function multiple times.

index matches the index used in [rtSelectorSetChild](#) on the host. [rtIntersectChild](#) is valid only within a selector visit program.

Parameters

| | | |
|-----------|--------------|--|
| <i>in</i> | <i>index</i> | Specifies the child to perform intersection on |
|-----------|--------------|--|

Return values

| | |
|-------------|-------------------|
| <i>void</i> | void return value |
|-------------|-------------------|

History

[rtIntersectChild](#) was introduced in OptiX 1.0.

See also [rtSelectorSetVisitProgram](#), [rtSelectorCreate](#), [rtTerminateRay](#)

3.6.3.5 `static __device__ bool rtPotentialIntersection (float tmin) [inline],[static]`

Determine whether a computed intersection is potentially valid.

Description

Reporting an intersection from a geometry program is a two-stage process. If the geometry program computes that the ray intersects the geometry, it will first call [rtPotentialIntersection](#). [rtPotentialIntersection](#) will determine whether the reported hit distance is within the valid interval associated with the ray, and return true if the intersection is valid. Subsequently, the geometry program will compute the attributes (normal, texture coordinates, etc.) associated with the intersection before calling [rtReportIntersection](#). When [rtReportIntersection](#) is called, the any-hit program associated with the material is called. If the any-hit program does not ignore the intersection then the *t* value will stand as the new closest intersection.

If [rtPotentialIntersection](#) returns true, then [rtReportIntersection](#) should **always** be called after computing the attributes. Furthermore, attributes variables should only be written after a successful return from [rtPotentialIntersection](#).

[rtPotentialIntersection](#) is passed the material index associated with the reported intersection. Objects with a single material should pass an index of zero.

[rtReportIntersection](#) and [rtPotentialIntersection](#) are valid only within a geometry intersection program.

Parameters

| | | |
|-----------|-------------|----------------------------------|
| <i>in</i> | <i>tmin</i> | t value of the ray to be checked |
|-----------|-------------|----------------------------------|

Return values

| | |
|-------------|--|
| <i>bool</i> | Returns whether the intersection is valid or not |
|-------------|--|

History

[rtPotentialIntersection](#) was introduced in OptiX 1.0.

See also [rtGeometrySetIntersectionProgram](#), [rtReportIntersection](#), [rtIgnoreIntersection](#)

3.6.3.6 `static __device__ void rtPrintExceptionDetails () [inline],[static]`

Print information on a caught exception.

Description

[rtGetExceptionCode](#) can be called from an exception program to provide information on the caught exception to the user. The function uses [rtPrintf](#) to output details depending on the type of the exception. It is necessary to have printing enabled using [rtContextSetPrintEnabled](#) for this function to have any effect.

Return values

| | |
|-------------|------------------|
| <i>void</i> | void return type |
|-------------|------------------|

History

[rtPrintExceptionDetails](#) was introduced in OptiX 1.1.

See also [rtContextSetExceptionEnabled](#), [rtContextGetExceptionEnabled](#), [rtContextSetExceptionProgram](#), [rtContextGetExceptionProgram](#), [rtContextSetPrintEnabled](#), [rtGetExceptionCode](#), [rtThrow](#), [rtPrintf](#)

```
3.6.3.7 static __device__ void rtPrintf ( const char * fmt ) [inline],[static]
```

Prints text to the standard output.

Description

`rtPrintf` is used to output text from within user programs. Arguments are passed as for the standard C `printf` function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using `rtPrintf` is accumulated in a buffer and printed to the standard output when `rtContextLaunch` finishes. The buffer size can be configured using `rtContextSetPrintBufferSize`. Output can optionally be restricted to certain launch indices using `rtContextSetPrintLaunchIndex`. Printing must be enabled using `rtContextSetPrintEnabled`, otherwise `rtPrintf` invocations will be silently ignored.

History

`rtPrintf` was introduced in OptiX 1.0.

See also `rtContextSetPrintEnabled`, `rtContextGetPrintEnabled`, `rtContextSetPrintBufferSize`, `rtContextGetPrintBufferSize`, `rtContextSetPrintLaunchIndex`, `rtContextSetPrintLaunchIndex`

```
3.6.3.8 template<typename T1> static __device__ void rtPrintf ( const char * fmt, T1 arg1 ) [inline],[static]
```

Prints text to the standard output.

Description

`rtPrintf` is used to output text from within user programs. Arguments are passed as for the standard C `printf` function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using `rtPrintf` is accumulated in a buffer and printed to the standard output when `rtContextLaunch` finishes. The buffer size can be configured using `rtContextSetPrintBufferSize`. Output can optionally be restricted to certain launch indices using `rtContextSetPrintLaunchIndex`. Printing must be enabled using `rtContextSetPrintEnabled`, otherwise `rtPrintf` invocations will be silently ignored.

History

`rtPrintf` was introduced in OptiX 1.0.

See also `rtContextSetPrintEnabled`, `rtContextGetPrintEnabled`, `rtContextSetPrintBufferSize`, `rtContextGetPrintBufferSize`, `rtContextSetPrintLaunchIndex`, `rtContextSetPrintLaunchIndex`

```
3.6.3.9 template<typename T1, typename T2> static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2 arg2 ) [inline],[static]
```

Prints text to the standard output.

Description

`rtPrintf` is used to output text from within user programs. Arguments are passed as for the standard C `printf` function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using `rtPrintf` is accumulated in a buffer and printed to the standard output when `rtContextLaunch` finishes. The buffer size can be configured using `rtContextSetPrintBufferSize`. Output can optionally be restricted to certain launch indices using `rtContextSetPrintLaunchIndex`. Printing must be enabled using `rtContextSetPrintEnabled`, otherwise `rtPrintf` invocations will be silently ignored.

History

`rtPrintf` was introduced in OptiX 1.0.

See also `rtContextSetPrintEnabled`, `rtContextGetPrintEnabled`, `rtContextSetPrintBufferSize`, `rtContextGetPrintBufferSize`, `rtContextSetPrintLaunchIndex`, `rtContextSetPrintLaunchIndex`

```
3.6.3.10 template<typename T1, typename T2, typename T3> static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2 arg2, T3 arg3 ) [inline],[static]
```

Prints text to the standard output.

Description

`rtPrintf` is used to output text from within user programs. Arguments are passed as for the standard C `printf` function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using `rtPrintf` is accumulated in a buffer and printed to the standard output when `rt-ContextLaunch` finishes. The buffer size can be configured using `rtContextSetPrintBufferSize`. Output can optionally be restricted to certain launch indices using `rtContextSetPrintLaunchIndex`. Printing must be enabled using `rt-ContextSetPrintEnabled`, otherwise `rtPrintf` invocations will be silently ignored.

History

`rtPrintf` was introduced in OptiX 1.0.

See also `rtContextSetPrintEnabled`, `rtContextGetPrintEnabled`, `rtContextSetPrintBufferSize`, `rtContextGetPrint-Buffer-Size`, `rtContextSetPrintLaunchIndex`, `rtContextSetPrintLaunchIndex`

```
3.6.3.11  template<typename T1 , typename T2 , typename T3 , typename T4 > static __device__ void rtPrintf ( const char *
          fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4 )  [inline], [static]
```

Prints text to the standard output.

Description

`rtPrintf` is used to output text from within user programs. Arguments are passed as for the standard C `printf` function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using `rtPrintf` is accumulated in a buffer and printed to the standard output when `rt-ContextLaunch` finishes. The buffer size can be configured using `rtContextSetPrintBufferSize`. Output can optionally be restricted to certain launch indices using `rtContextSetPrintLaunchIndex`. Printing must be enabled using `rt-ContextSetPrintEnabled`, otherwise `rtPrintf` invocations will be silently ignored.

History

`rtPrintf` was introduced in OptiX 1.0.

See also `rtContextSetPrintEnabled`, `rtContextGetPrintEnabled`, `rtContextSetPrintBufferSize`, `rtContextGetPrint-Buffer-Size`, `rtContextSetPrintLaunchIndex`, `rtContextSetPrintLaunchIndex`

```
3.6.3.12  template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 > static __device__ void rtPrintf (
          const char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5 )  [inline], [static]
```

Prints text to the standard output.

Description

`rtPrintf` is used to output text from within user programs. Arguments are passed as for the standard C `printf` function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using `rtPrintf` is accumulated in a buffer and printed to the standard output when `rt-ContextLaunch` finishes. The buffer size can be configured using `rtContextSetPrintBufferSize`. Output can optionally be restricted to certain launch indices using `rtContextSetPrintLaunchIndex`. Printing must be enabled using `rt-ContextSetPrintEnabled`, otherwise `rtPrintf` invocations will be silently ignored.

History

`rtPrintf` was introduced in OptiX 1.0.

See also `rtContextSetPrintEnabled`, `rtContextGetPrintEnabled`, `rtContextSetPrintBufferSize`, `rtContextGetPrint-Buffer-Size`, `rtContextSetPrintLaunchIndex`, `rtContextSetPrintLaunchIndex`

```
3.6.3.13  template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 > static __device__
          void rtPrintf ( const char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6 )  [inline], [static]
```

Prints text to the standard output.

Description

`rtPrintf` is used to output text from within user programs. Arguments are passed as for the standard C `printf` function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using `rtPrintf` is accumulated in a buffer and printed to the standard output when `rt-ContextLaunch` finishes. The buffer size can be configured using `rtContextSetPrintBufferSize`. Output can optionally

be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

History

[rtPrintf](#) was introduced in OptiX 1.0.

See also [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.6.3.14 template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 >
        static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7 )
        [inline],[static]
```

Prints text to the standard output.

Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

History

[rtPrintf](#) was introduced in OptiX 1.0.

See also [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.6.3.15 template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 ,
        typename T8 > static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6
        arg6, T7 arg7, T8 arg8 ) [inline],[static]
```

Prints text to the standard output.

Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

History

[rtPrintf](#) was introduced in OptiX 1.0.

See also [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.6.3.16 template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 ,
        typename T8 , typename T9 > static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4,
        T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9 ) [inline],[static]
```

Prints text to the standard output.

Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rt-](#)

[ContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

History

[rtPrintf](#) was introduced in OptiX 1.0.

See also [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.6.3.17  template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 ,
          typename T8 , typename T9 , typename T10 > static __device__ void rtPrintf ( const char * fmt, T1 arg1, T2 arg2, T3
          arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10 )  [inline],[static]
```

Prints text to the standard output.

Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

History

[rtPrintf](#) was introduced in OptiX 1.0.

See also [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.6.3.18  template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 ,
          typename T8 , typename T9 , typename T10 , typename T11 > static __device__ void rtPrintf ( const char * fmt, T1
          arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10, T11 arg11 )  [inline],
          [static]
```

Prints text to the standard output.

Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

History

[rtPrintf](#) was introduced in OptiX 1.0.

See also [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

```
3.6.3.19  template<typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 ,
          typename T8 , typename T9 , typename T10 , typename T11 , typename T12 > static __device__ void rtPrintf ( const
          char * fmt, T1 arg1, T2 arg2, T3 arg3, T4 arg4, T5 arg5, T6 arg6, T7 arg7, T8 arg8, T9 arg9, T10 arg10, T11 arg11,
          T12 arg12 )  [inline],[static]
```

Prints text to the standard output.

Description

[rtPrintf](#) is used to output text from within user programs. Arguments are passed as for the standard C *printf* function, and the same format strings are employed. The only exception is the "%s" format specifier, which will generate an error if used. Text printed using [rtPrintf](#) is accumulated in a buffer and printed to the standard output when [rtContextLaunch](#) finishes. The buffer size can be configured using [rtContextSetPrintBufferSize](#). Output can optionally

be restricted to certain launch indices using [rtContextSetPrintLaunchIndex](#). Printing must be enabled using [rtContextSetPrintEnabled](#), otherwise [rtPrintf](#) invocations will be silently ignored.

History

[rtPrintf](#) was introduced in OptiX 1.0.

See also [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextSetPrintLaunchIndex](#)

3.6.3.20 `static __device__ bool rtReportIntersection (unsigned int material) [inline],[static]`

Report an intersection with the current object and the specified material.

Description

[rtReportIntersection](#) reports an intersection of the current ray with the current object, and specifies the material associated with the intersection. [rtReportIntersection](#) should only be used in conjunction with [rtPotentialIntersection](#) as described in [rtPotentialIntersection](#).

Parameters

| | | |
|-----------------|------------------------------|---|
| <code>in</code> | <code><i>material</i></code> | Material associated with the intersection |
|-----------------|------------------------------|---|

Return values

| | |
|-------------------|--|
| <code>bool</code> | return value, this is set to <i>false</i> if the intersection is, for some reason, ignored |
|-------------------|--|

History

[rtReportIntersection](#) was introduced in OptiX 1.0.

See also [rtPotentialIntersection](#), [rtIgnoreIntersection](#)

3.6.3.21 `static __device__ void rtTerminateRay () [inline],[static]`

Terminate traversal associated with the current ray.

Description

[rtTerminateRay](#) causes the traversal associated with the current ray to immediately terminate. After termination, the closest-hit program associated with the ray will be called. This function does not return, so values affecting the per-ray data should be applied before calling [rtTerminateRay](#). [rtTerminateRay](#) is valid only within an any-hit program. The value of `rtIntersectionDistance` is undefined when [rtTerminateRay](#) is used.

Return values

| | |
|-------------------|-------------------|
| <code>void</code> | void return value |
|-------------------|-------------------|

History

[rtTerminateRay](#) was introduced in OptiX 1.0.

See also [rtIgnoreIntersection](#), [rtPotentialIntersection](#)

3.6.3.22 `static __device__ void rtThrow (unsigned int code) [inline],[static]`

Throw a user exception.

Description

[rtThrow](#) is used to trigger user defined exceptions which behave like built-in exceptions. That is, upon invocation, ray processing for the current launch index is immediately aborted and the corresponding exception program is executed. [rtThrow](#) does not return.

The *code* passed as argument must be within the range reserved for user exceptions, which starts at [RT_EXCEPTION_USER](#) (`0x400`) and ends at `0xFFFF`. The code can be queried within the exception program using [rtGetExceptionCode](#).

[rtThrow](#) may be called from within any program type except exception programs. Calls to [rtThrow](#) will be silently ignored unless user exceptions are enabled using [rtContextSetExceptionEnabled](#).

History

[rtThrow](#) was introduced in OptiX 1.1.

See also [rtContextSetExceptionEnabled](#), [rtContextGetExceptionEnabled](#), [rtContextSetExceptionProgram](#), [rtContextGetExceptionProgram](#), [rtGetExceptionCode](#), [rtPrintExceptionDetails](#)

3.6.3.23 `template<class T> static __device__ void rtTrace (rtObject topNode, optix::Ray ray, T & prd) [inline], [static]`

Traces a ray.

Description

[rtTrace](#) traces *ray* against object *topNode*. A reference to *prd*, the per-ray data, will be passed to all of the closest-hit and any-hit programs that are executed during this invocation of trace. *topNode* must refer to an OptiX object of type [RTgroup](#), [RTselector](#), [RTgeometrygroup](#) or [RTtransform](#).

Parameters

| | | |
|----|----------------|--|
| in | <i>topNode</i> | Top node object where to start the traversal |
| in | <i>ray</i> | Ray to be traced |
| in | <i>prd</i> | Per-ray custom data |

Return values

| | |
|-------------|-------------------|
| <i>void</i> | void return value |
|-------------|-------------------|

History

[rtTrace](#) was introduced in OptiX 1.0.

See also [rtObject Ray](#)

3.6.3.24 `static __device__ float3 rtTransformNormal (RTtransformkind kind, const float3 & n) [inline], [static]`

Apply the current transformation to a normal.

Description

[rtTransformNormal](#) transforms *n* as a normal using the current active transformation stack (the inverse transpose). During traversal, intersection and any-hit programs, the current ray will be located in object space. During ray generation, closest-hit and miss programs, the current ray will be located in world space. This function can be used to transform values between object and world space.

kind is an enumerated value that can be either [RT_OBJECT_TO_WORLD](#) or [RT_WORLD_TO_OBJECT](#) and must be a constant literal. For ray generation and miss programs, the transform will always be the identity transform. For traversal, intersection, any-hit and closest-hit programs, the transform will be dependent on the set of active transform nodes for the current state.

Parameters

| | | |
|----|-------------|-----------------------|
| in | <i>kind</i> | Type of the transform |
| in | <i>n</i> | Normal to transform |

Return values

| | |
|---------------|--------------------|
| <i>float3</i> | Transformed normal |
|---------------|--------------------|

History

[rtTransformNormal](#) was introduced in OptiX 1.0.

See also [rtTransformCreate](#), [rtTransformPoint](#), [rtTransformVector](#)

3.6.3.25 `static __device__ float3 rtTransformPoint (RTtransformkind kind, const float3 & p) [inline], [static]`

Apply the current transformation to a point.

Description

[rtTransformPoint](#) transforms p as a point using the current active transformation stack. During traversal, intersection and any-hit programs, the current ray will be located in object space. During ray generation, closest-hit and miss programs, the current ray will be located in world space. This function can be used to transform the ray origin and other points between object and world space.

kind is an enumerated value that can be either [RT_OBJECT_TO_WORLD](#) or [RT_WORLD_TO_OBJECT](#) and must be a constant literal. For ray generation and miss programs, the transform will always be the identity transform. For traversal, intersection, any-hit and closest-hit programs, the transform will be dependent on the set of active transform nodes for the current state.

Parameters

| | | |
|-----------|-------------|-----------------------|
| <i>in</i> | <i>kind</i> | Type of the transform |
| <i>in</i> | <i>p</i> | Point to transform |

Return values

| | |
|---------------|-------------------|
| <i>float3</i> | Transformed point |
|---------------|-------------------|

History

[rtTransformPoint](#) was introduced in OptiX 1.0.

See also [rtTransformCreate](#), [rtTransformVector](#), [rtTransformNormal](#)

```
3.6.3.26 static __device__ float3 rtTransformVector ( RTtransformkind kind, const float3 & v ) [inline],
[static]
```

Apply the current transformation to a vector.

Description

[rtTransformVector](#) transforms v as a vector using the current active transformation stack. During traversal, intersection and any-hit programs, the current ray will be located in object space. During ray generation, closest-hit and miss programs, the current ray will be located in world space. This function can be used to transform the ray direction and other vectors between object and world space.

kind is an enumerated value that can be either [RT_OBJECT_TO_WORLD](#) or [RT_WORLD_TO_OBJECT](#) and must be a constant literal. For ray generation and miss programs, the transform will always be the identity transform. For traversal, intersection, any-hit and closest-hit programs, the transform will be dependent on the set of active transform nodes for the current state.

Parameters

| | | |
|-----------|-------------|-----------------------|
| <i>in</i> | <i>kind</i> | Type of the transform |
| <i>in</i> | <i>v</i> | Vector to transform |

Return values

| | |
|---------------|--------------------|
| <i>float3</i> | Transformed vector |
|---------------|--------------------|

History

[rtTransformVector](#) was introduced in OptiX 1.0.

See also [rtTransformCreate](#), [rtTransformPoint](#), [rtTransformNormal](#)

3.7 optix_gl_interop.h File Reference

3.7.1 Detailed Description

OptiX public API declarations GLInterop.

Author

NVIDIA Corporation OptiX public API declarations for GL interoperability

Typedefs

- typedef void * **HGPUNV**

Functions

- [RTresult](#) RTAPI [rtBufferCreateFromGLBO](#) ([RTcontext](#) context, unsigned int bufferdesc, unsigned int glld, [RTbuffer](#) *buffer)
- [RTresult](#) RTAPI [rtTextureSamplerCreateFromGLImage](#) ([RTcontext](#) context, unsigned int glld, [RTgltarget](#) target, [RTtexturesampler](#) *textureSampler)
- [RTresult](#) RTAPI [rtBufferGetGLBOld](#) ([RTbuffer](#) buffer, unsigned int *glld)
- [RTresult](#) RTAPI [rtTextureSamplerGetGLImageId](#) ([RTtexturesampler](#) textureSampler, unsigned int *glld)
- [RTresult](#) RTAPI [rtBufferGLRegister](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtBufferGLUnregister](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtTextureSamplerGLRegister](#) ([RTtexturesampler](#) textureSampler)
- [RTresult](#) RTAPI [rtTextureSamplerGLUnregister](#) ([RTtexturesampler](#) textureSampler)
- [RTresult](#) RTAPI [rtDeviceGetWGLDevice](#) (int *device, HGPUNV gpu)

3.7.2 Function Documentation

3.7.2.1 [RTresult](#) RTAPI [rtBufferCreateFromGLBO](#) ([RTcontext](#) context, unsigned int bufferdesc, unsigned int glld, [RTbuffer](#) * buffer)

Creates a new buffer object from an OpenGL buffer object.

Description

[rtBufferCreateFromGLBO](#) allocates and returns a handle to a new buffer object in *buffer associated with context. Supported OpenGL buffer types are:

- Pixel Buffer Objects
- Vertex Buffer Objects

These buffers can be used to share data with OpenGL; changes of the content in buffer, either done by OpenGL or OptiX, will be reflected automatically in both APIs. If the size, or format, of an OpenGL buffer is changed, appropriate OptiX calls have to be used to update buffer accordingly. OptiX keeps only a reference to OpenGL data, when buffer is destroyed, the state of the gl_id object is unaltered.

The type of this buffer is specified by one of the following values in bufferdesc:

- [RT_BUFFER_INPUT](#)
- [RT_BUFFER_OUTPUT](#)
- [RT_BUFFER_INPUT_OUTPUT](#)

The type values are used to specify the direction of data flow from the host to the OptiX devices. [RT_BUFFER_INPUT](#) specifies that the host may only write to the buffer and the device may only read from the buffer. [RT_BUFFER_OUTPUT](#) specifies the opposite, read only access on the host and write only access on the device. Devices and the host may read and write from buffers of type [RT_BUFFER_INPUT_OUTPUT](#). Reading or writing to a buffer of the incorrect type (e.g., the host writing to a buffer of type [RT_BUFFER_OUTPUT](#)) is undefined.

Flags can be used to optimize data transfers between the host and it's devices. Currently no flags are supported for interop buffers.

Parameters

| | | |
|-----|-------------------|---|
| in | <i>context</i> | The context to create the buffer in |
| in | <i>bufferdesc</i> | Bitwise <i>or</i> combination of the <i>type</i> and <i>flags</i> of the new buffer |
| in | <i>glId</i> | The OpenGL image object resource handle for use in OptiX |
| out | <i>buffer</i> | The return handle for the buffer object |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferCreateFromGLBO](#) was introduced in OptiX 1.0.

See also [rtBufferCreate](#), [rtBufferDestroy](#)

3.7.2.2 RTresult RTAPI rtBufferGetGLBOId (RTbuffer *buffer*, unsigned int * *glId*)

Gets the OpenGL Buffer Object ID associated with this buffer.

Description

[rtBufferGetGLBOId](#) stores the OpenGL buffer object id in *gl_id* if *buffer* was created with [rtBufferCreateFromGLBO](#). If *buffer* was not created from an OpenGL Buffer Object *gl_id* will be set to 0.

Parameters

| | | |
|----|---------------|--|
| in | <i>buffer</i> | The buffer to be queried for its OpenGL buffer object id |
| in | <i>glId</i> | The return handle for the id |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferGetGLBOId](#) was introduced in OptiX 1.0.

See also [rtBufferCreateFromGLBO](#)

3.7.2.3 RTresult RTAPI rtBufferGLRegister (RTbuffer *buffer*)

Declares an OpenGL buffer as immutable and accessible by OptiX.

Description

Once registered, properties like the size of the original GL buffer cannot be modified anymore. Calls to the corresponding GL functions will return with an error code. However, the buffer data of the GL buffer can still be read and written by the appropriate GL commands. Returns [RT_ERROR_RESOURCE_ALREADY_REGISTERED](#) if *buffer* is already registered. A buffer object must be registered in order to be used by OptiX. If a buffer object is not registered [RT_ERROR_INVALID_VALUE](#) will be returned. An OptiX buffer in a registered state can be unregistered via [rtBufferGLRegister](#).

Parameters

| | | |
|-----------------|---------------------|----------------------------------|
| <code>in</code> | <code>buffer</code> | The handle for the buffer object |
|-----------------|---------------------|----------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_RESOURCE_ALREADY_REGISTERED](#)

History

[rtBufferGLRegister](#) was introduced in OptiX 2.0.

See also [rtBufferCreateFromGLBO](#), [rtBufferGLUnregister](#)

3.7.2.4 `RTresult RTAPI rtBufferGLUnregister (RTbuffer buffer)`

Declares an OpenGL buffer as mutable and inaccessible by OptiX.

Description

Once unregistered, properties like the size of the original GL buffer can be changed. As long as a buffer object is unregistered, OptiX will not be able to access the data and calls will fail with [RT_ERROR_INVALID_VALUE](#). Returns [RT_ERROR_RESOURCE_NOT_REGISTERED](#) if `buffer` is already unregistered. An OptiX buffer in an unregistered state can be registered to OptiX again via [rtBufferGLRegister](#).

Parameters

| | | |
|-----------------|---------------------|----------------------------------|
| <code>in</code> | <code>buffer</code> | The handle for the buffer object |
|-----------------|---------------------|----------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_RESOURCE_NOT_REGISTERED](#)

History

[rtBufferGLUnregister](#) was introduced in OptiX 2.0.

See also [rtBufferCreateFromGLBO](#), [rtBufferGLRegister](#)

3.7.2.5 `RTresult RTAPI rtDeviceGetWGLDevice (int * device, HGPUNV gpu)`

returns the OptiX device number associated with the specified GPU

Description

[rtDeviceGetWGLDevice](#) returns in `device` the OptiX device ID of the GPU represented by `gpu`. `gpu` is returned from `WGL_NV_gpu_affinity`, an OpenGL extension. This enables OptiX to create a context on the same GPU that OpenGL commands will be sent to, improving OpenGL interoperation efficiency.

Parameters

| | | |
|-----|--------|--|
| out | device | A handle to the memory location where the OptiX device ordinal associated with <i>gpu</i> will be stored |
| in | gpu | A handle to a GPU as returned from the <i>WGL_NV_gpu_affinity</i> OpenGL extension |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtDeviceGetWGLDevice](#) was introduced in OptiX 1.0.

See also [rtDeviceGetDeviceCount](#), *WGL_NV_gpu_affinity*

3.7.2.6 RTresult RTAPI rtTextureSamplerCreateFromGLImage (RTcontext *context*, unsigned int *glId*, RTgltarget *target*, RTtexturesampler * *textureSampler*)

Creates a new texture sampler object from an OpenGL image.

Description

[rtTextureSamplerCreateFromGLImage](#) allocates and returns a handle to a new texture sampler object in * *textureSampler* associated with *context*. If the allocated size of the GL texture is 0, [RT_ERROR_MEMORY_ALLOCATION_FAILED](#) will be returned. Supported OpenGL image types are:

Renderbuffers

- GL_TEXTURE_2D
- GL_TEXTURE_2D_RECT
- GL_TEXTURE_3D

These types are reflected by *target*:

- [RT_TARGET_GL_RENDER_BUFFER](#)
- [RT_TARGET_GL_TEXTURE_1D](#)
- [RT_TARGET_GL_TEXTURE_2D](#)
- [RT_TARGET_GL_TEXTURE_RECTANGLE](#)
- [RT_TARGET_GL_TEXTURE_3D](#)
- [RT_TARGET_GL_TEXTURE_1D_ARRAY](#)
- [RT_TARGET_GL_TEXTURE_2D_ARRAY](#)
- [RT_TARGET_GL_TEXTURE_CUBE_MAP](#)
- [RT_TARGET_GL_TEXTURE_CUBE_MAP_ARRAY](#)

Supported attachment points for renderbuffers are:

- GL_COLOR_ATTACHMENT<NUM>

These texture samplers can be used to share data with OpenGL; changes of the content and size of *texturesampler* done by OpenGL will be reflected automatically in OptiX. Currently texture sampler data are read only in OptiX programs. OptiX keeps only a reference to OpenGL data, when *texturesampler* is destroyed, the state of the *gl_id* image is unaltered.

The array size and number of mipmap levels can't be changed for texture samplers that encapsulate a GL image. Furthermore no buffer objects can be queried.

Currently OptiX supports only a limited number of internal OpenGL texture formats. Texture formats with an internal type of float, e.g. *GL_RGBA32F*, and many integer formats are supported. Depth formats as well as multisample buffers are also currently not supported. Please refer to the InteropTypes section for a complete list of supported texture formats.

Parameters

| | | |
|-----|-----------------------|--|
| in | <i>context</i> | The context to create the buffer in |
| in | <i>glId</i> | The OpenGL image object resource handle for use in OptiX |
| in | <i>target</i> | The OpenGL target |
| out | <i>textureSampler</i> | The return handle for the texture sampler object |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTextureSamplerCreateFromGLImage](#) was introduced in OptiX 2.0.

See also [rtTextureSamplerCreate](#), [rtTextureSamplerDestroy](#)

3.7.2.7 RTresult RTAPI rtTextureSamplerGetGLImageId (RTtexturesampler *textureSampler*, unsigned int * *glId*)

Gets the OpenGL image object id associated with this texture sampler.

Description

[rtTextureSamplerGetGLImageId](#) stores the OpenGL image object id in *gl_id* if *textureSampler* was created with [rtTextureSamplerCreateFromGLImage](#). If *textureSampler* was not created from an OpenGL image object *gl_id* will be set to 0.

Parameters

| | | |
|----|-----------------------|---|
| in | <i>textureSampler</i> | The texture sampler to be queried for its OpenGL buffer object id |
| in | <i>glId</i> | The return handle for the id |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTextureSamplerGetGLImageId](#) was introduced in OptiX 2.0.

See also [rtTextureSamplerCreateFromGLImage](#)

3.7.2.8 RTresult RTAPI rtTextureSamplerGLRegister (RTtexturesampler textureSampler)

Declares an OpenGL texture as immutable and accessible by OptiX.

Description

Registers an OpenGL texture as accessible by OptiX. Once registered, properties like the size of the original GL texture cannot be modified anymore. Calls to the corresponding GL functions will return with an error code. However, the pixel data of the GL texture can still be read and written by the appropriate GL commands. Returns [RT_ERROR_RESOURCE_ALREADY_REGISTERED](#) if *textureSampler* is already registered. A texture sampler must be registered in order to be used by OptiX. Otherwise, [RT_ERROR_INVALID_VALUE](#) is returned. An OptiX texture sampler in a registered state can be unregistered via [rtTextureSamplerGLUnregister](#).

Parameters

| | | |
|-----------|-----------------------|-----------------------------------|
| <i>in</i> | <i>textureSampler</i> | The handle for the texture object |
|-----------|-----------------------|-----------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_RESOURCE_ALREADY_REGISTERED](#)

History

[rtTextureSamplerGLRegister](#) was introduced in OptiX 2.0.

See also [rtTextureSamplerCreateFromGLImage](#), [rtTextureSamplerGLUnregister](#)

3.7.2.9 RTresult RTAPI rtTextureSamplerGLUnregister (RTtexturesampler textureSampler)

Declares an OpenGL texture as mutable and inaccessible by OptiX.

Description

Once unregistered, properties like the size of the original GL texture can be changed. As long as a texture is unregistered, OptiX will not be able to access the pixel data and calls will fail with [RT_ERROR_INVALID_VALUE](#). Returns [RT_ERROR_RESOURCE_NOT_REGISTERED](#) if *textureSampler* is already unregistered. An OptiX texture sampler in an unregistered state can be registered to OptiX again via [rtTextureSamplerGLRegister](#).

Parameters

| | | |
|-----------|-----------------------|-----------------------------------|
| <i>in</i> | <i>textureSampler</i> | The handle for the texture object |
|-----------|-----------------------|-----------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_RESOURCE_NOT_REGISTERED](#)

History

[rtTextureSamplerGLUnregister](#) was introduced in OptiX 2.0.

See also [rtTextureSamplerCreateFromGLImage](#), [rtTextureSamplerGLRegister](#)

3.8 optix_host.h File Reference

3.8.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Host side

Macros

- `#define RTAPI __declspec(dllimport)`

Typedefs

- `typedef unsigned int RTsize`
- `typedef struct RTacceleration_api * RTacceleration`
- `typedef struct RTbuffer_api * RTbuffer`
- `typedef struct RTcontext_api * RTcontext`
- `typedef struct RTgeometry_api * RTgeometry`
- `typedef struct
RTgeometryinstance_api * RTgeometryinstance`
- `typedef struct
RTgeometrygroup_api * RTgeometrygroup`
- `typedef struct RTgroup_api * RTgroup`
- `typedef struct RTmaterial_api * RTmaterial`
- `typedef struct RTprogram_api * RTprogram`
- `typedef struct RTselector_api * RTselector`
- `typedef struct
RTtexturesampler_api * RTtexturesampler`
- `typedef struct RTtransform_api * RTtransform`
- `typedef struct RTvariable_api * RTvariable`
- `typedef void * RObject`
- `typedef struct RTremotedevice_api * RTremotedevice`
- `typedef int(* RTtimeoutcallback)(void)`

Functions

- `RTresult RTAPI rtGetVersion (unsigned int *version)`
- `RTresult RTAPI rtDeviceGetDeviceCount (unsigned int *count)`
- `RTresult RTAPI rtDeviceGetAttribute (int ordinal, RTdeviceattribute attrib, RTsize size, void *p)`
- `RTresult RTAPI rtVariableSetObject (RTvariable v, RObject object)`
- `RTresult RTAPI rtVariableSetUserData (RTvariable v, RTsize size, const void *ptr)`
- `RTresult RTAPI rtVariableGetObject (RTvariable v, RObject *object)`
- `RTresult RTAPI rtVariableGetUserData (RTvariable v, RTsize size, void *ptr)`
- `RTresult RTAPI rtVariableGetName (RTvariable v, const char **name_return)`
- `RTresult RTAPI rtVariableGetAnnotation (RTvariable v, const char **annotation_return)`
- `RTresult RTAPI rtVariableGetType (RTvariable v, RObjecttype *type_return)`
- `RTresult RTAPI rtVariableGetContext (RTvariable v, RTcontext *context)`
- `RTresult RTAPI rtVariableGetSize (RTvariable v, RTsize *size)`
- `RTresult RTAPI rtContextCreate (RTcontext *context)`
- `RTresult RTAPI rtContextDestroy (RTcontext context)`
- `RTresult RTAPI rtContextValidate (RTcontext context)`
- `void RTAPI rtContextGetErrorString (RTcontext context, RTresult code, const char **return_string)`

- [RTresult](#) RTAPI [rtContextSetAttribute](#) ([RTcontext](#) context, [RTcontextattribute](#) attrib, [RTsize](#) size, void *p)
- [RTresult](#) RTAPI [rtContextGetAttribute](#) ([RTcontext](#) context, [RTcontextattribute](#) attrib, [RTsize](#) size, void *p)
- [RTresult](#) RTAPI [rtContextSetDevices](#) ([RTcontext](#) context, unsigned int count, const int *devices)
- [RTresult](#) RTAPI [rtContextGetDevices](#) ([RTcontext](#) context, int *devices)
- [RTresult](#) RTAPI [rtContextGetDeviceCount](#) ([RTcontext](#) context, unsigned int *count)
- [RTresult](#) RTAPI [rtContextSetRemoteDevice](#) ([RTcontext](#) context, [RTremotedevice](#) remote_dev)
- [RTresult](#) RTAPI [rtContextSetStackSize](#) ([RTcontext](#) context, [RTsize](#) stack_size_bytes)
- [RTresult](#) RTAPI [rtContextGetStackSize](#) ([RTcontext](#) context, [RTsize](#) *stack_size_bytes)
- [RTresult](#) RTAPI [rtContextSetTimeoutCallback](#) ([RTcontext](#) context, [RTtimeoutcallback](#) callback, double min_polling_seconds)
- [RTresult](#) RTAPI [rtContextSetEntryPointCount](#) ([RTcontext](#) context, unsigned int num_entry_points)
- [RTresult](#) RTAPI [rtContextGetEntryPointCount](#) ([RTcontext](#) context, unsigned int *num_entry_points)
- [RTresult](#) RTAPI [rtContextSetRayGenerationProgram](#) ([RTcontext](#) context, unsigned int entry_point_index, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtContextGetRayGenerationProgram](#) ([RTcontext](#) context, unsigned int entry_point_index, [RTprogram](#) *program)
- [RTresult](#) RTAPI [rtContextSetExceptionProgram](#) ([RTcontext](#) context, unsigned int entry_point_index, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtContextGetExceptionProgram](#) ([RTcontext](#) context, unsigned int entry_point_index, [RTprogram](#) *program)
- [RTresult](#) RTAPI [rtContextSetExceptionEnabled](#) ([RTcontext](#) context, [RTexception](#) exception, int enabled)
- [RTresult](#) RTAPI [rtContextGetExceptionEnabled](#) ([RTcontext](#) context, [RTexception](#) exception, int *enabled)
- [RTresult](#) RTAPI [rtContextSetRayTypeCount](#) ([RTcontext](#) context, unsigned int num_ray_types)
- [RTresult](#) RTAPI [rtContextGetRayTypeCount](#) ([RTcontext](#) context, unsigned int *num_ray_types)
- [RTresult](#) RTAPI [rtContextSetMissProgram](#) ([RTcontext](#) context, unsigned int ray_type_index, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtContextGetMissProgram](#) ([RTcontext](#) context, unsigned int ray_type_index, [RTprogram](#) *program)
- [RTresult](#) RTAPI [rtContextGetTextureSamplerFromId](#) ([RTcontext](#) context, int sampler_id, [RTtexturesampler](#) *sampler)
- [RTresult](#) RTAPI [rtContextCompile](#) ([RTcontext](#) context)
- [RTresult](#) RTAPI [rtContextLaunch1D](#) ([RTcontext](#) context, unsigned int entry_point_index, [RTsize](#) image_width)
- [RTresult](#) RTAPI [rtContextLaunch2D](#) ([RTcontext](#) context, unsigned int entry_point_index, [RTsize](#) image_width, [RTsize](#) image_height)
- [RTresult](#) RTAPI [rtContextLaunch3D](#) ([RTcontext](#) context, unsigned int entry_point_index, [RTsize](#) image_width, [RTsize](#) image_height, [RTsize](#) image_depth)
- [RTresult](#) RTAPI [rtContextGetRunningState](#) ([RTcontext](#) context, int *running)
- [RTresult](#) RTAPI [rtContextLaunchProgressive2D](#) ([RTcontext](#) context, unsigned int entry_index, [RTsize](#) width, [RTsize](#) height, unsigned int max_subframes)
- [RTresult](#) RTAPI [rtContextStopProgressive](#) ([RTcontext](#) context)
- [RTresult](#) RTAPI [rtContextSetPrintEnabled](#) ([RTcontext](#) context, int enabled)
- [RTresult](#) RTAPI [rtContextGetPrintEnabled](#) ([RTcontext](#) context, int *enabled)
- [RTresult](#) RTAPI [rtContextSetPrintBufferSize](#) ([RTcontext](#) context, [RTsize](#) buffer_size_bytes)
- [RTresult](#) RTAPI [rtContextGetPrintBufferSize](#) ([RTcontext](#) context, [RTsize](#) *buffer_size_bytes)
- [RTresult](#) RTAPI [rtContextSetPrintLaunchIndex](#) ([RTcontext](#) context, int x, int y, int z)
- [RTresult](#) RTAPI [rtContextGetPrintLaunchIndex](#) ([RTcontext](#) context, int *x, int *y, int *z)
- [RTresult](#) RTAPI [rtContextDeclareVariable](#) ([RTcontext](#) context, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtContextQueryVariable](#) ([RTcontext](#) context, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtContextRemoveVariable](#) ([RTcontext](#) context, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtContextGetVariableCount](#) ([RTcontext](#) context, unsigned int *count)
- [RTresult](#) RTAPI [rtContextGetVariable](#) ([RTcontext](#) context, unsigned int index, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtProgramCreateFromPTXString](#) ([RTcontext](#) context, const char *ptx, const char *program_name, [RTprogram](#) *program)
- [RTresult](#) RTAPI [rtProgramCreateFromPTXFile](#) ([RTcontext](#) context, const char *filename, const char *program_name, [RTprogram](#) *program)

- [RTresult](#) RTAPI [rtProgramDestroy](#) ([RTprogram](#) program)
- [RTresult](#) RTAPI [rtProgramValidate](#) ([RTprogram](#) program)
- [RTresult](#) RTAPI [rtProgramGetContext](#) ([RTprogram](#) program, [RTcontext](#) *context)
- [RTresult](#) RTAPI [rtProgramDeclareVariable](#) ([RTprogram](#) program, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtProgramQueryVariable](#) ([RTprogram](#) program, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtProgramRemoveVariable](#) ([RTprogram](#) program, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtProgramGetVariableCount](#) ([RTprogram](#) program, unsigned int *count)
- [RTresult](#) RTAPI [rtProgramGetVariable](#) ([RTprogram](#) program, unsigned int index, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtProgramGetId](#) ([RTprogram](#) program, int *program_id)
- [RTresult](#) RTAPI [rtContextGetProgramFromId](#) ([RTcontext](#) context, int program_id, [RTprogram](#) *program)
- [RTresult](#) RTAPI [rtGroupCreate](#) ([RTcontext](#) context, [RTgroup](#) *group)
- [RTresult](#) RTAPI [rtGroupDestroy](#) ([RTgroup](#) group)
- [RTresult](#) RTAPI [rtGroupValidate](#) ([RTgroup](#) group)
- [RTresult](#) RTAPI [rtGroupGetContext](#) ([RTgroup](#) group, [RTcontext](#) *context)
- [RTresult](#) RTAPI [rtGroupSetAcceleration](#) ([RTgroup](#) group, [RTacceleration](#) acceleration)
- [RTresult](#) RTAPI [rtGroupGetAcceleration](#) ([RTgroup](#) group, [RTacceleration](#) *acceleration)
- [RTresult](#) RTAPI [rtGroupSetChildCount](#) ([RTgroup](#) group, unsigned int count)
- [RTresult](#) RTAPI [rtGroupGetChildCount](#) ([RTgroup](#) group, unsigned int *count)
- [RTresult](#) RTAPI [rtGroupSetChild](#) ([RTgroup](#) group, unsigned int index, [RTobject](#) child)
- [RTresult](#) RTAPI [rtGroupGetChild](#) ([RTgroup](#) group, unsigned int index, [RTobject](#) *child)
- [RTresult](#) RTAPI [rtGroupGetChildType](#) ([RTgroup](#) group, unsigned int index, [RTobjecttype](#) *type)
- [RTresult](#) RTAPI [rtSelectorCreate](#) ([RTcontext](#) context, [RTselector](#) *selector)
- [RTresult](#) RTAPI [rtSelectorDestroy](#) ([RTselector](#) selector)
- [RTresult](#) RTAPI [rtSelectorValidate](#) ([RTselector](#) selector)
- [RTresult](#) RTAPI [rtSelectorGetContext](#) ([RTselector](#) selector, [RTcontext](#) *context)
- [RTresult](#) RTAPI [rtSelectorSetVisitProgram](#) ([RTselector](#) selector, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtSelectorGetVisitProgram](#) ([RTselector](#) selector, [RTprogram](#) *program)
- [RTresult](#) RTAPI [rtSelectorSetChildCount](#) ([RTselector](#) selector, unsigned int count)
- [RTresult](#) RTAPI [rtSelectorGetChildCount](#) ([RTselector](#) selector, unsigned int *count)
- [RTresult](#) RTAPI [rtSelectorSetChild](#) ([RTselector](#) selector, unsigned int index, [RTobject](#) child)
- [RTresult](#) RTAPI [rtSelectorGetChild](#) ([RTselector](#) selector, unsigned int index, [RTobject](#) *child)
- [RTresult](#) RTAPI [rtSelectorGetChildType](#) ([RTselector](#) selector, unsigned int index, [RTobjecttype](#) *type)
- [RTresult](#) RTAPI [rtSelectorDeclareVariable](#) ([RTselector](#) selector, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtSelectorQueryVariable](#) ([RTselector](#) selector, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtSelectorRemoveVariable](#) ([RTselector](#) selector, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtSelectorGetVariableCount](#) ([RTselector](#) selector, unsigned int *count)
- [RTresult](#) RTAPI [rtSelectorGetVariable](#) ([RTselector](#) selector, unsigned int index, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtTransformCreate](#) ([RTcontext](#) context, [RTtransform](#) *transform)
- [RTresult](#) RTAPI [rtTransformDestroy](#) ([RTtransform](#) transform)
- [RTresult](#) RTAPI [rtTransformValidate](#) ([RTtransform](#) transform)
- [RTresult](#) RTAPI [rtTransformGetContext](#) ([RTtransform](#) transform, [RTcontext](#) *context)
- [RTresult](#) RTAPI [rtTransformSetMatrix](#) ([RTtransform](#) transform, int transpose, const float *matrix, const float *inverse_matrix)
- [RTresult](#) RTAPI [rtTransformGetMatrix](#) ([RTtransform](#) transform, int transpose, float *matrix, float *inverse_matrix)
- [RTresult](#) RTAPI [rtTransformSetChild](#) ([RTtransform](#) transform, [RTobject](#) child)
- [RTresult](#) RTAPI [rtTransformGetChild](#) ([RTtransform](#) transform, [RTobject](#) *child)
- [RTresult](#) RTAPI [rtTransformGetChildType](#) ([RTtransform](#) transform, [RTobjecttype](#) *type)
- [RTresult](#) RTAPI [rtGeometryGroupCreate](#) ([RTcontext](#) context, [RTgeometrygroup](#) *geometrygroup)
- [RTresult](#) RTAPI [rtGeometryGroupDestroy](#) ([RTgeometrygroup](#) geometrygroup)
- [RTresult](#) RTAPI [rtGeometryGroupValidate](#) ([RTgeometrygroup](#) geometrygroup)
- [RTresult](#) RTAPI [rtGeometryGroupGetContext](#) ([RTgeometrygroup](#) geometrygroup, [RTcontext](#) *context)
- [RTresult](#) RTAPI [rtGeometryGroupSetAcceleration](#) ([RTgeometrygroup](#) geometrygroup, [RTacceleration](#) acceleration)

- [RTresult](#) RTAPI [rtGeometryGroupGetAcceleration](#) ([RTgeometrygroup](#) geometrygroup, [RTacceleration](#) *acceleration)
- [RTresult](#) RTAPI [rtGeometryGroupSetChildCount](#) ([RTgeometrygroup](#) geometrygroup, unsigned int count)
- [RTresult](#) RTAPI [rtGeometryGroupGetChildCount](#) ([RTgeometrygroup](#) geometrygroup, unsigned int *count)
- [RTresult](#) RTAPI [rtGeometryGroupSetChild](#) ([RTgeometrygroup](#) geometrygroup, unsigned int index, [RTgeometryinstance](#) geometryinstance)
- [RTresult](#) RTAPI [rtGeometryGroupGetChild](#) ([RTgeometrygroup](#) geometrygroup, unsigned int index, [RTgeometryinstance](#) *geometryinstance)
- [RTresult](#) RTAPI [rtAccelerationCreate](#) ([RTcontext](#) context, [RTacceleration](#) *acceleration)
- [RTresult](#) RTAPI [rtAccelerationDestroy](#) ([RTacceleration](#) acceleration)
- [RTresult](#) RTAPI [rtAccelerationValidate](#) ([RTacceleration](#) acceleration)
- [RTresult](#) RTAPI [rtAccelerationGetContext](#) ([RTacceleration](#) acceleration, [RTcontext](#) *context)
- [RTresult](#) RTAPI [rtAccelerationSetBuilder](#) ([RTacceleration](#) acceleration, const char *builder)
- [RTresult](#) RTAPI [rtAccelerationGetBuilder](#) ([RTacceleration](#) acceleration, const char **return_string)
- [RTresult](#) RTAPI [rtAccelerationSetTraverser](#) ([RTacceleration](#) acceleration, const char *traverser)
- [RTresult](#) RTAPI [rtAccelerationGetTraverser](#) ([RTacceleration](#) acceleration, const char **return_string)
- [RTresult](#) RTAPI [rtAccelerationSetProperty](#) ([RTacceleration](#) acceleration, const char *name, const char *value)
- [RTresult](#) RTAPI [rtAccelerationGetProperty](#) ([RTacceleration](#) acceleration, const char *name, const char **return_string)
- [RTresult](#) RTAPI [rtAccelerationGetDataSize](#) ([RTacceleration](#) acceleration, [RTsize](#) *size)
- [RTresult](#) RTAPI [rtAccelerationGetData](#) ([RTacceleration](#) acceleration, void *data)
- [RTresult](#) RTAPI [rtAccelerationSetData](#) ([RTacceleration](#) acceleration, const void *data, [RTsize](#) size)
- [RTresult](#) RTAPI [rtAccelerationMarkDirty](#) ([RTacceleration](#) acceleration)
- [RTresult](#) RTAPI [rtAccelerationIsDirty](#) ([RTacceleration](#) acceleration, int *dirty)
- [RTresult](#) RTAPI [rtGeometryInstanceCreate](#) ([RTcontext](#) context, [RTgeometryinstance](#) *geometryinstance)
- [RTresult](#) RTAPI [rtGeometryInstanceDestroy](#) ([RTgeometryinstance](#) geometryinstance)
- [RTresult](#) RTAPI [rtGeometryInstanceValidate](#) ([RTgeometryinstance](#) geometryinstance)
- [RTresult](#) RTAPI [rtGeometryInstanceGetContext](#) ([RTgeometryinstance](#) geometryinstance, [RTcontext](#) *context)
- [RTresult](#) RTAPI [rtGeometryInstanceSetGeometry](#) ([RTgeometryinstance](#) geometryinstance, [RTgeometry](#) geometry)
- [RTresult](#) RTAPI [rtGeometryInstanceGetGeometry](#) ([RTgeometryinstance](#) geometryinstance, [RTgeometry](#) *geometry)
- [RTresult](#) RTAPI [rtGeometryInstanceSetMaterialCount](#) ([RTgeometryinstance](#) geometryinstance, unsigned int count)
- [RTresult](#) RTAPI [rtGeometryInstanceGetMaterialCount](#) ([RTgeometryinstance](#) geometryinstance, unsigned int *count)
- [RTresult](#) RTAPI [rtGeometryInstanceSetMaterial](#) ([RTgeometryinstance](#) geometryinstance, unsigned int index, [RTmaterial](#) material)
- [RTresult](#) RTAPI [rtGeometryInstanceGetMaterial](#) ([RTgeometryinstance](#) geometryinstance, unsigned int index, [RTmaterial](#) *material)
- [RTresult](#) RTAPI [rtGeometryInstanceDeclareVariable](#) ([RTgeometryinstance](#) geometryinstance, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtGeometryInstanceQueryVariable](#) ([RTgeometryinstance](#) geometryinstance, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtGeometryInstanceRemoveVariable](#) ([RTgeometryinstance](#) geometryinstance, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtGeometryInstanceGetVariableCount](#) ([RTgeometryinstance](#) geometryinstance, unsigned int *count)
- [RTresult](#) RTAPI [rtGeometryInstanceGetVariable](#) ([RTgeometryinstance](#) geometryinstance, unsigned int index, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtGeometryCreate](#) ([RTcontext](#) context, [RTgeometry](#) *geometry)
- [RTresult](#) RTAPI [rtGeometryDestroy](#) ([RTgeometry](#) geometry)
- [RTresult](#) RTAPI [rtGeometryValidate](#) ([RTgeometry](#) geometry)
- [RTresult](#) RTAPI [rtGeometryGetContext](#) ([RTgeometry](#) geometry, [RTcontext](#) *context)
- [RTresult](#) RTAPI [rtGeometrySetPrimitiveCount](#) ([RTgeometry](#) geometry, unsigned int num_primitives)

- [RTresult](#) RTAPI [rtGeometryGetPrimitiveCount](#) ([RTgeometry](#) geometry, unsigned int *num_primitives)
- [RTresult](#) RTAPI [rtGeometrySetPrimitiveIndexOffset](#) ([RTgeometry](#) geometry, unsigned int index_offset)
- [RTresult](#) RTAPI [rtGeometryGetPrimitiveIndexOffset](#) ([RTgeometry](#) geometry, unsigned int *index_offset)
- [RTresult](#) RTAPI [rtGeometrySetBoundingBoxProgram](#) ([RTgeometry](#) geometry, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtGeometryGetBoundingBoxProgram](#) ([RTgeometry](#) geometry, [RTprogram](#) *program)
- [RTresult](#) RTAPI [rtGeometrySetIntersectionProgram](#) ([RTgeometry](#) geometry, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtGeometryGetIntersectionProgram](#) ([RTgeometry](#) geometry, [RTprogram](#) *program)
- [RTresult](#) RTAPI [rtGeometryMarkDirty](#) ([RTgeometry](#) geometry)
- [RTresult](#) RTAPI [rtGeometryIsDirty](#) ([RTgeometry](#) geometry, int *dirty)
- [RTresult](#) RTAPI [rtGeometryDeclareVariable](#) ([RTgeometry](#) geometry, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtGeometryQueryVariable](#) ([RTgeometry](#) geometry, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtGeometryRemoveVariable](#) ([RTgeometry](#) geometry, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtGeometryGetVariableCount](#) ([RTgeometry](#) geometry, unsigned int *count)
- [RTresult](#) RTAPI [rtGeometryGetVariable](#) ([RTgeometry](#) geometry, unsigned int index, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtMaterialCreate](#) ([RTcontext](#) context, [RTmaterial](#) *material)
- [RTresult](#) RTAPI [rtMaterialDestroy](#) ([RTmaterial](#) material)
- [RTresult](#) RTAPI [rtMaterialValidate](#) ([RTmaterial](#) material)
- [RTresult](#) RTAPI [rtMaterialGetContext](#) ([RTmaterial](#) material, [RTcontext](#) *context)
- [RTresult](#) RTAPI [rtMaterialSetClosestHitProgram](#) ([RTmaterial](#) material, unsigned int ray_type_index, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtMaterialGetClosestHitProgram](#) ([RTmaterial](#) material, unsigned int ray_type_index, [RTprogram](#) *program)
- [RTresult](#) RTAPI [rtMaterialSetAnyHitProgram](#) ([RTmaterial](#) material, unsigned int ray_type_index, [RTprogram](#) program)
- [RTresult](#) RTAPI [rtMaterialGetAnyHitProgram](#) ([RTmaterial](#) material, unsigned int ray_type_index, [RTprogram](#) *program)
- [RTresult](#) RTAPI [rtMaterialDeclareVariable](#) ([RTmaterial](#) material, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtMaterialQueryVariable](#) ([RTmaterial](#) material, const char *name, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtMaterialRemoveVariable](#) ([RTmaterial](#) material, [RTvariable](#) v)
- [RTresult](#) RTAPI [rtMaterialGetVariableCount](#) ([RTmaterial](#) material, unsigned int *count)
- [RTresult](#) RTAPI [rtMaterialGetVariable](#) ([RTmaterial](#) material, unsigned int index, [RTvariable](#) *v)
- [RTresult](#) RTAPI [rtTextureSamplerCreate](#) ([RTcontext](#) context, [RTtexturesampler](#) *texturesampler)
- [RTresult](#) RTAPI [rtTextureSamplerDestroy](#) ([RTtexturesampler](#) texturesampler)
- [RTresult](#) RTAPI [rtTextureSamplerValidate](#) ([RTtexturesampler](#) texturesampler)
- [RTresult](#) RTAPI [rtTextureSamplerGetContext](#) ([RTtexturesampler](#) texturesampler, [RTcontext](#) *context)
- [RTresult](#) RTAPI [rtTextureSamplerSetMipLevelCount](#) ([RTtexturesampler](#) texturesampler, unsigned int num_mip_levels)
- [RTresult](#) RTAPI [rtTextureSamplerGetMipLevelCount](#) ([RTtexturesampler](#) texturesampler, unsigned int *num_mip_levels)
- [RTresult](#) RTAPI [rtTextureSamplerSetArraySize](#) ([RTtexturesampler](#) texturesampler, unsigned int num_textures_in_array)
- [RTresult](#) RTAPI [rtTextureSamplerGetArraySize](#) ([RTtexturesampler](#) texturesampler, unsigned int *num_textures_in_array)
- [RTresult](#) RTAPI [rtTextureSamplerSetWrapMode](#) ([RTtexturesampler](#) texturesampler, unsigned int dimension, [RTwrapmode](#) wrapmode)
- [RTresult](#) RTAPI [rtTextureSamplerGetWrapMode](#) ([RTtexturesampler](#) texturesampler, unsigned int dimension, [RTwrapmode](#) *wrapmode)
- [RTresult](#) RTAPI [rtTextureSamplerSetFilteringModes](#) ([RTtexturesampler](#) texturesampler, [RTfiltermode](#) minification, [RTfiltermode](#) magnification, [RTfiltermode](#) mipmapping)
- [RTresult](#) RTAPI [rtTextureSamplerGetFilteringModes](#) ([RTtexturesampler](#) texturesampler, [RTfiltermode](#) *minification, [RTfiltermode](#) *magnification, [RTfiltermode](#) *mipmapping)
- [RTresult](#) RTAPI [rtTextureSamplerSetMaxAnisotropy](#) ([RTtexturesampler](#) texturesampler, float value)
- [RTresult](#) RTAPI [rtTextureSamplerGetMaxAnisotropy](#) ([RTtexturesampler](#) texturesampler, float *value)
- [RTresult](#) RTAPI [rtTextureSamplerSetMipLevelClamp](#) ([RTtexturesampler](#) texturesampler, float minLevel, float maxLevel)

- [RTresult](#) RTAPI [rtTextureSamplerGetMipLevelClamp](#) ([RTtexturesampler](#) texturesampler, float *minLevel, float *maxLevel)
- [RTresult](#) RTAPI [rtTextureSamplerSetMipLevelBias](#) ([RTtexturesampler](#) texturesampler, float value)
- [RTresult](#) RTAPI [rtTextureSamplerGetMipLevelBias](#) ([RTtexturesampler](#) texturesampler, float *value)
- [RTresult](#) RTAPI [rtTextureSamplerSetReadMode](#) ([RTtexturesampler](#) texturesampler, [RTtexturereadmode](#) readmode)
- [RTresult](#) RTAPI [rtTextureSamplerGetReadMode](#) ([RTtexturesampler](#) texturesampler, [RTtexturereadmode](#) *readmode)
- [RTresult](#) RTAPI [rtTextureSamplerSetIndexingMode](#) ([RTtexturesampler](#) texturesampler, [RTtextureindexmode](#) indexmode)
- [RTresult](#) RTAPI [rtTextureSamplerGetIndexingMode](#) ([RTtexturesampler](#) texturesampler, [RTtextureindexmode](#) *indexmode)
- [RTresult](#) RTAPI [rtTextureSamplerSetBuffer](#) ([RTtexturesampler](#) texturesampler, unsigned int deprecated0, unsigned int deprecated1, [RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtTextureSamplerGetBuffer](#) ([RTtexturesampler](#) texturesampler, unsigned int deprecated0, unsigned int deprecated1, [RTbuffer](#) *buffer)
- [RTresult](#) RTAPI [rtTextureSamplerGetId](#) ([RTtexturesampler](#) texturesampler, int *texture_id)
- [RTresult](#) RTAPI [rtBufferCreate](#) ([RTcontext](#) context, unsigned int bufferdesc, [RTbuffer](#) *buffer)
- [RTresult](#) RTAPI [rtBufferDestroy](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtBufferValidate](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtBufferGetContext](#) ([RTbuffer](#) buffer, [RTcontext](#) *context)
- [RTresult](#) RTAPI [rtBufferSetFormat](#) ([RTbuffer](#) buffer, [RTformat](#) format)
- [RTresult](#) RTAPI [rtBufferGetFormat](#) ([RTbuffer](#) buffer, [RTformat](#) *format)
- [RTresult](#) RTAPI [rtBufferSetElementSize](#) ([RTbuffer](#) buffer, [RTsize](#) size_of_element)
- [RTresult](#) RTAPI [rtBufferGetElementSize](#) ([RTbuffer](#) buffer, [RTsize](#) *size_of_element)
- [RTresult](#) RTAPI [rtBufferSetSize1D](#) ([RTbuffer](#) buffer, [RTsize](#) width)
- [RTresult](#) RTAPI [rtBufferGetSize1D](#) ([RTbuffer](#) buffer, [RTsize](#) *width)
- [RTresult](#) RTAPI [rtBufferSetSize2D](#) ([RTbuffer](#) buffer, [RTsize](#) width, [RTsize](#) height)
- [RTresult](#) RTAPI [rtBufferGetSize2D](#) ([RTbuffer](#) buffer, [RTsize](#) *width, [RTsize](#) *height)
- [RTresult](#) RTAPI [rtBufferSetSize3D](#) ([RTbuffer](#) buffer, [RTsize](#) width, [RTsize](#) height, [RTsize](#) depth)
- [RTresult](#) RTAPI [rtBufferSetMipLevelCount](#) ([RTbuffer](#) buffer, unsigned int levels)
- [RTresult](#) RTAPI [rtBufferGetSize3D](#) ([RTbuffer](#) buffer, [RTsize](#) *width, [RTsize](#) *height, [RTsize](#) *depth)
- [RTresult](#) RTAPI [rtBufferGetMipLevelSize1D](#) ([RTbuffer](#) buffer, unsigned int level, [RTsize](#) *width)
- [RTresult](#) RTAPI [rtBufferGetMipLevelSize2D](#) ([RTbuffer](#) buffer, unsigned int level, [RTsize](#) *width, [RTsize](#) *height)
- [RTresult](#) RTAPI [rtBufferGetMipLevelSize3D](#) ([RTbuffer](#) buffer, unsigned int level, [RTsize](#) *width, [RTsize](#) *height, [RTsize](#) *depth)
- [RTresult](#) RTAPI [rtBufferSetSizev](#) ([RTbuffer](#) buffer, unsigned int dimensionality, const [RTsize](#) *dims)
- [RTresult](#) RTAPI [rtBufferGetSizev](#) ([RTbuffer](#) buffer, unsigned int dimensionality, [RTsize](#) *dims)
- [RTresult](#) RTAPI [rtBufferGetDimensionality](#) ([RTbuffer](#) buffer, unsigned int *dimensionality)
- [RTresult](#) RTAPI [rtBufferGetMipLevelCount](#) ([RTbuffer](#) buffer, unsigned int *level)
- [RTresult](#) RTAPI [rtBufferMap](#) ([RTbuffer](#) buffer, void **user_pointer)
- [RTresult](#) RTAPI [rtBufferUnmap](#) ([RTbuffer](#) buffer)
- [RTresult](#) RTAPI [rtBufferMapEx](#) ([RTbuffer](#) buffer, unsigned int map_flags, unsigned int level, void *user_owned, void **optix_owned)
- [RTresult](#) RTAPI [rtBufferUnmapEx](#) ([RTbuffer](#) buffer, unsigned int level)
- [RTresult](#) RTAPI [rtBufferGetId](#) ([RTbuffer](#) buffer, int *buffer_id)
- [RTresult](#) RTAPI [rtContextGetBufferFromId](#) ([RTcontext](#) context, int buffer_id, [RTbuffer](#) *buffer)
- [RTresult](#) RTAPI [rtBufferGetProgressiveUpdateReady](#) ([RTbuffer](#) buffer, int *ready, unsigned int *subframe_count, unsigned int *max_subframes)
- [RTresult](#) RTAPI [rtBufferBindProgressiveStream](#) ([RTbuffer](#) stream, [RTbuffer](#) source)
- [RTresult](#) RTAPI [rtBufferSetAttribute](#) ([RTbuffer](#) buffer, [RTbufferattribute](#) attrib, [RTsize](#) size, void *p)
- [RTresult](#) RTAPI [rtBufferGetAttribute](#) ([RTbuffer](#) buffer, [RTbufferattribute](#) attrib, [RTsize](#) size, void *p)
- [RTresult](#) RTAPI [rtRemoteDeviceCreate](#) (const char *url, const char *username, const char *password, [RTremotedevice](#) *remote_dev)

- [RTresult](#) RTAPI [rtRemoteDeviceDestroy](#) ([RTremotedevice](#) remote_dev)
- [RTresult](#) RTAPI [rtRemoteDeviceGetAttribute](#) ([RTremotedevice](#) remote_dev, [RTremotedeviceattribute](#) attrib, [RTsize](#) size, void *p)
- [RTresult](#) RTAPI [rtRemoteDeviceReserve](#) ([RTremotedevice](#) remote_dev, unsigned int num_nodes, unsigned int configuration)
- [RTresult](#) RTAPI [rtRemoteDeviceRelease](#) ([RTremotedevice](#) remote_dev)

- [RTresult](#) RTAPI [rtVariableSet1f](#) ([RTvariable](#) v, float f1)
- [RTresult](#) RTAPI [rtVariableSet2f](#) ([RTvariable](#) v, float f1, float f2)
- [RTresult](#) RTAPI [rtVariableSet3f](#) ([RTvariable](#) v, float f1, float f2, float f3)
- [RTresult](#) RTAPI [rtVariableSet4f](#) ([RTvariable](#) v, float f1, float f2, float f3, float f4)
- [RTresult](#) RTAPI [rtVariableSet1fv](#) ([RTvariable](#) v, const float *f)
- [RTresult](#) RTAPI [rtVariableSet2fv](#) ([RTvariable](#) v, const float *f)
- [RTresult](#) RTAPI [rtVariableSet3fv](#) ([RTvariable](#) v, const float *f)
- [RTresult](#) RTAPI [rtVariableSet4fv](#) ([RTvariable](#) v, const float *f)
- [RTresult](#) RTAPI [rtVariableSet1i](#) ([RTvariable](#) v, int i1)
- [RTresult](#) RTAPI [rtVariableSet2i](#) ([RTvariable](#) v, int i1, int i2)
- [RTresult](#) RTAPI [rtVariableSet3i](#) ([RTvariable](#) v, int i1, int i2, int i3)
- [RTresult](#) RTAPI [rtVariableSet4i](#) ([RTvariable](#) v, int i1, int i2, int i3, int i4)
- [RTresult](#) RTAPI [rtVariableSet1iv](#) ([RTvariable](#) v, const int *i)
- [RTresult](#) RTAPI [rtVariableSet2iv](#) ([RTvariable](#) v, const int *i)
- [RTresult](#) RTAPI [rtVariableSet3iv](#) ([RTvariable](#) v, const int *i)
- [RTresult](#) RTAPI [rtVariableSet4iv](#) ([RTvariable](#) v, const int *i)
- [RTresult](#) RTAPI [rtVariableSet1ui](#) ([RTvariable](#) v, unsigned int u1)
- [RTresult](#) RTAPI [rtVariableSet2ui](#) ([RTvariable](#) v, unsigned int u1, unsigned int u2)
- [RTresult](#) RTAPI [rtVariableSet3ui](#) ([RTvariable](#) v, unsigned int u1, unsigned int u2, unsigned int u3)
- [RTresult](#) RTAPI [rtVariableSet4ui](#) ([RTvariable](#) v, unsigned int u1, unsigned int u2, unsigned int u3, unsigned int u4)
- [RTresult](#) RTAPI [rtVariableSet1uiv](#) ([RTvariable](#) v, const unsigned int *u)
- [RTresult](#) RTAPI [rtVariableSet2uiv](#) ([RTvariable](#) v, const unsigned int *u)
- [RTresult](#) RTAPI [rtVariableSet3uiv](#) ([RTvariable](#) v, const unsigned int *u)
- [RTresult](#) RTAPI [rtVariableSet4uiv](#) ([RTvariable](#) v, const unsigned int *u)
- [RTresult](#) RTAPI [rtVariableSetMatrix2x2fv](#) ([RTvariable](#) v, int transpose, const float *m)
- [RTresult](#) RTAPI [rtVariableSetMatrix2x3fv](#) ([RTvariable](#) v, int transpose, const float *m)
- [RTresult](#) RTAPI [rtVariableSetMatrix2x4fv](#) ([RTvariable](#) v, int transpose, const float *m)
- [RTresult](#) RTAPI [rtVariableSetMatrix3x2fv](#) ([RTvariable](#) v, int transpose, const float *m)
- [RTresult](#) RTAPI [rtVariableSetMatrix3x3fv](#) ([RTvariable](#) v, int transpose, const float *m)
- [RTresult](#) RTAPI [rtVariableSetMatrix3x4fv](#) ([RTvariable](#) v, int transpose, const float *m)
- [RTresult](#) RTAPI [rtVariableSetMatrix4x2fv](#) ([RTvariable](#) v, int transpose, const float *m)
- [RTresult](#) RTAPI [rtVariableSetMatrix4x3fv](#) ([RTvariable](#) v, int transpose, const float *m)
- [RTresult](#) RTAPI [rtVariableSetMatrix4x4fv](#) ([RTvariable](#) v, int transpose, const float *m)

- [RTresult](#) RTAPI [rtVariableGet1f](#) ([RTvariable](#) v, float *f1)
- [RTresult](#) RTAPI [rtVariableGet2f](#) ([RTvariable](#) v, float *f1, float *f2)
- [RTresult](#) RTAPI [rtVariableGet3f](#) ([RTvariable](#) v, float *f1, float *f2, float *f3)
- [RTresult](#) RTAPI [rtVariableGet4f](#) ([RTvariable](#) v, float *f1, float *f2, float *f3, float *f4)
- [RTresult](#) RTAPI [rtVariableGet1fv](#) ([RTvariable](#) v, float *f)
- [RTresult](#) RTAPI [rtVariableGet2fv](#) ([RTvariable](#) v, float *f)
- [RTresult](#) RTAPI [rtVariableGet3fv](#) ([RTvariable](#) v, float *f)
- [RTresult](#) RTAPI [rtVariableGet4fv](#) ([RTvariable](#) v, float *f)
- [RTresult](#) RTAPI [rtVariableGet1i](#) ([RTvariable](#) v, int *i1)
- [RTresult](#) RTAPI [rtVariableGet2i](#) ([RTvariable](#) v, int *i1, int *i2)
- [RTresult](#) RTAPI [rtVariableGet3i](#) ([RTvariable](#) v, int *i1, int *i2, int *i3)
- [RTresult](#) RTAPI [rtVariableGet4i](#) ([RTvariable](#) v, int *i1, int *i2, int *i3, int *i4)
- [RTresult](#) RTAPI [rtVariableGet1iv](#) ([RTvariable](#) v, int *i)

- [RTresult](#) RTAPI [rtVariableGet2iv](#) ([RTvariable](#) v, int *i)
- [RTresult](#) RTAPI [rtVariableGet3iv](#) ([RTvariable](#) v, int *i)
- [RTresult](#) RTAPI [rtVariableGet4iv](#) ([RTvariable](#) v, int *i)
- [RTresult](#) RTAPI [rtVariableGet1ui](#) ([RTvariable](#) v, unsigned int *u1)
- [RTresult](#) RTAPI [rtVariableGet2ui](#) ([RTvariable](#) v, unsigned int *u1, unsigned int *u2)
- [RTresult](#) RTAPI [rtVariableGet3ui](#) ([RTvariable](#) v, unsigned int *u1, unsigned int *u2, unsigned int *u3)
- [RTresult](#) RTAPI [rtVariableGet4ui](#) ([RTvariable](#) v, unsigned int *u1, unsigned int *u2, unsigned int *u3, unsigned int *u4)
- [RTresult](#) RTAPI [rtVariableGet1uiv](#) ([RTvariable](#) v, unsigned int *u)
- [RTresult](#) RTAPI [rtVariableGet2uiv](#) ([RTvariable](#) v, unsigned int *u)
- [RTresult](#) RTAPI [rtVariableGet3uiv](#) ([RTvariable](#) v, unsigned int *u)
- [RTresult](#) RTAPI [rtVariableGet4uiv](#) ([RTvariable](#) v, unsigned int *u)
- [RTresult](#) RTAPI [rtVariableGetMatrix2x2fv](#) ([RTvariable](#) v, int transpose, float *m)
- [RTresult](#) RTAPI [rtVariableGetMatrix2x3fv](#) ([RTvariable](#) v, int transpose, float *m)
- [RTresult](#) RTAPI [rtVariableGetMatrix2x4fv](#) ([RTvariable](#) v, int transpose, float *m)
- [RTresult](#) RTAPI [rtVariableGetMatrix3x2fv](#) ([RTvariable](#) v, int transpose, float *m)
- [RTresult](#) RTAPI [rtVariableGetMatrix3x3fv](#) ([RTvariable](#) v, int transpose, float *m)
- [RTresult](#) RTAPI [rtVariableGetMatrix3x4fv](#) ([RTvariable](#) v, int transpose, float *m)
- [RTresult](#) RTAPI [rtVariableGetMatrix4x2fv](#) ([RTvariable](#) v, int transpose, float *m)
- [RTresult](#) RTAPI [rtVariableGetMatrix4x3fv](#) ([RTvariable](#) v, int transpose, float *m)
- [RTresult](#) RTAPI [rtVariableGetMatrix4x4fv](#) ([RTvariable](#) v, int transpose, float *m)

3.8.2 Typedef Documentation

3.8.2.1 typedef struct RTacceleration_api* RTacceleration

Opaque type to handle Acceleration Structures - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.2 typedef struct RTbuffer_api* RTbuffer

Opaque type to handle Buffers - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.3 typedef struct RTcontext_api* RTcontext

Opaque type to handle Contexts - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.4 typedef struct RTgeometry_api* RTgeometry

Opaque type to handle Geometry - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.5 typedef struct RTgeometrygroup_api* RTgeometrygroup

Opaque type to handle Geometry Group - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.6 typedef struct RTgeometryinstance_api* RTgeometryinstance

Opaque type to handle Geometry Instance - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.7 typedef struct RTgroup_api* RTgroup

Opaque type to handle Group - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.8 typedef struct RTmaterial_api* RTmaterial

Opaque type to handle Material - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.9 typedef void* RObject

Opaque type to handle Object - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.10 typedef struct RTprogram_api* RTprogram

Opaque type to handle Program - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.11 typedef struct RTremotedevice_api* RTremotedevice

Opaque type to handle RemoteDevice - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.12 typedef struct RTselector_api* RTselector

Opaque type to handle Selector - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.13 typedef struct RTtexturesampler_api* RTtexturesampler

Opaque type to handle Texture Sampler - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.14 typedef int(* RTtimeoutcallback)(void)

Callback signature for use with rtContextSetTimeoutCallback. Return 1 to ask for abort, 0 to continue.

3.8.2.15 typedef struct RTtransform_api* RTtransform

Opaque type to handle Transform - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.2.16 typedef struct RTvariable_api* RTvariable

Opaque type to handle Variable - Note that the *_api type should never be used directly. Only the typedef target name will be guaranteed to remain unchanged

3.8.3 Function Documentation**3.8.3.1 RTresult RTAPI rtAccelerationCreate (RTcontext context, RTacceleration * acceleration)**

Creates a new acceleration structure.

Description

[rtAccelerationCreate](#) creates a new ray tracing acceleration structure within a context. An acceleration structure is used by attaching it to a group or geometry group by calling [rtGroupSetAcceleration](#) or [rtGeometryGroupSetAcceleration](#). Note that an acceleration structure can be shared by attaching it to multiple groups or geometry groups if the underlying geometric structures are the same, see [rtGroupSetAcceleration](#) and [rtGeometryGroupSetAcceleration](#) for more details. A newly created acceleration structure is initially in dirty state. Sets *acceleration to the handle of a newly created acceleration structure within context. Returns [RT_ERROR_INVALID_VALUE](#) if acceleration is NULL.

Parameters

| | | |
|-----|---------------------|---|
| in | <i>context</i> | Specifies a context within which to create a new acceleration structure |
| out | <i>acceleration</i> | Returns the newly created acceleration structure |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtAccelerationCreate](#) was introduced in OptiX 1.0.

See also [rtAccelerationDestroy](#), [rtContextCreate](#), [rtAccelerationMarkDirty](#), [rtAccelerationIsDirty](#), [rtGroupSetAcceleration](#), [rtGeometryGroupSetAcceleration](#)

3.8.3.2 RTresult RTAPI rtAccelerationDestroy (RTacceleration *acceleration*)

Destroys an acceleration structure object.

Description

[rtAccelerationDestroy](#) removes *acceleration* from its context and deletes it. *acceleration* should be a value returned by [rtAccelerationCreate](#). After the call, *acceleration* is no longer a valid handle.

Parameters

| | | |
|----|---------------------|---|
| in | <i>acceleration</i> | Handle of the acceleration structure to destroy |
|----|---------------------|---|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtAccelerationDestroy](#) was introduced in OptiX 1.0.

See also [rtAccelerationCreate](#)

3.8.3.3 RTresult RTAPI rtAccelerationGetBuilder (RTacceleration *acceleration*, const char ** *return_string*)

Query the current builder from an acceleration structure.

Description

[rtAccelerationGetBuilder](#) returns the name of the builder currently used in the acceleration structure *acceleration*. If no builder has been set for *acceleration*, an empty string is returned. *return_string* will be set to point to the returned string. The memory *return_string* points to will be valid until the next API call that returns a string.

Parameters

| | | |
|----|---------------------|-----------------------------------|
| in | <i>acceleration</i> | The acceleration structure handle |
|----|---------------------|-----------------------------------|

| | | |
|-----|----------------------|----------------------|
| out | <i>return_string</i> | Return string buffer |
|-----|----------------------|----------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtAccelerationGetBuilder](#) was introduced in OptiX 1.0.

See also [rtAccelerationSetBuilder](#)

3.8.3.4 RTresult RTAPI rtAccelerationGetContext (RTacceleration *acceleration*, RTcontext * *context*)

Returns the context associated with an acceleration structure.

Description

[rtAccelerationGetContext](#) queries an acceleration structure for its associated context. The context handle is returned in **context*.

Parameters

| | | |
|-----|---------------------|--|
| in | <i>acceleration</i> | The acceleration structure handle |
| out | <i>context</i> | Returns the context associated with the acceleration structure |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtAccelerationGetContext](#) was introduced in OptiX 1.0.

See also [rtAccelerationCreate](#)

3.8.3.5 RTresult RTAPI rtAccelerationGetData (RTacceleration *acceleration*, void * *data*)

Deprecated in OptiX 4.0. Should not be called.

3.8.3.6 RTresult RTAPI rtAccelerationGetDataSize (RTacceleration *acceleration*, RTsize * *size*)

Deprecated in OptiX 4.0. Should not be called.

3.8.3.7 RTresult RTAPI rtAccelerationGetProperty (RTacceleration *acceleration*, const char * *name*, const char ** *return_string*)

Queries an acceleration structure property.

Description

[rtAccelerationGetProperty](#) returns the value of the acceleration structure property *name*. See [rtAccelerationSetProperty](#) for a list of supported properties. If the property name is not found, an empty string is returned. *return_string* will be set to point to the returned string. The memory *return_string* points to will be valid until the next API call that returns a string.

Parameters

| | | |
|-----|----------------------|--|
| in | <i>acceleration</i> | The acceleration structure handle |
| in | <i>name</i> | The name of the property to be queried |
| out | <i>return_string</i> | Return string buffer |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtAccelerationGetProperty](#) was introduced in OptiX 1.0.

See also [rtAccelerationSetProperty](#), [rtAccelerationSetBuilder](#),

3.8.3.8 **RTresult RTAPI rtAccelerationGetTraverser (RTacceleration *acceleration*, const char ** *return_string*)**

Deprecated in OptiX 4.0.

3.8.3.9 **RTresult RTAPI rtAccelerationIsDirty (RTacceleration *acceleration*, int * *dirty*)**

Returns the dirty flag of an acceleration structure.

Description

[rtAccelerationIsDirty](#) returns whether the acceleration structure is currently marked dirty. If the flag is set, a nonzero value will be returned in **dirty*. Otherwise, zero is returned.

Any acceleration structure which is marked dirty will be rebuilt on a call to one of the [rtContextLaunch](#) functions, and its dirty flag will be reset.

An acceleration structure which is not marked dirty will never be rebuilt, even if associated groups, geometry, properties, or any other values have changed.

Initially after creation, acceleration structures are marked dirty.

Parameters

| | | |
|-----|---------------------|-----------------------------------|
| in | <i>acceleration</i> | The acceleration structure handle |
| out | <i>dirty</i> | Returned dirty flag |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtAccelerationIsDirty](#) was introduced in OptiX 1.0.

See also [rtAccelerationMarkDirty](#), [rtContextLaunch](#)

3.8.3.10 **RTresult RTAPI rtAccelerationMarkDirty (RTacceleration *acceleration*)**

Marks an acceleration structure as dirty.

Description

[rtAccelerationMarkDirty](#) sets the dirty flag for *acceleration*.

Any acceleration structure which is marked dirty will be rebuilt on a call to one of the [rtContextLaunch](#) functions, and its dirty flag will be reset.

An acceleration structure which is not marked dirty will never be rebuilt, even if associated groups, geometry, properties, or any other values have changed.

Initially after creation, acceleration structures are marked dirty.

Parameters

| | | |
|----|---------------------|-----------------------------------|
| in | <i>acceleration</i> | The acceleration structure handle |
|----|---------------------|-----------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtAccelerationMarkDirty](#) was introduced in OptiX 1.0.

See also [rtAccelerationsIsDirty](#), [rtContextLaunch](#)

3.8.3.11 RTresult RTAPI rtAccelerationSetBuilder (RTacceleration *acceleration*, const char * *builder*)

Specifies the builder to be used for an acceleration structure.

Description

[rtAccelerationSetBuilder](#) specifies the method used to construct the ray tracing acceleration structure represented by *acceleration*. A builder must be set for the acceleration structure to pass validation. The current builder can be changed at any time, including after a call to [rtContextLaunch](#). In this case, data previously computed for the acceleration structure is invalidated and the acceleration will be marked dirty.

builder can take one of the following values:

- "NoAccel": Specifies that no acceleration structure is explicitly built. Traversal linearly loops through the list of primitives to intersect. This can be useful e.g. for higher level groups with only few children, where managing a more complex structure introduces unnecessary overhead.
- "Bvh": A standard bounding volume hierarchy, useful for most types of graph levels and geometry. Medium build speed, good ray tracing performance.
- "Sbvh": A high quality BVH variant for maximum ray tracing performance. Slower build speed and slightly higher memory footprint than "Bvh".
- "Trbvh": High quality similar to Sbvh but with fast build performance. The Trbvh builder uses about 2.5 times the size of the final BVH for scratch space. A CPU-based Trbvh builder that does not have the memory constraints is available. OptiX includes an optional automatic fallback to the CPU version when out of GPU memory. Please refer to the Programming Guide for more details.
- "MedianBvh": Deprecated in OptiX 4.0. This builder is now internally remapped to Trbvh.
- "Lbvh": Deprecated in OptiX 4.0. This builder is now internally remapped to Trbvh.
- "TriangleKdTree": Deprecated in OptiX 4.0. This builder is now internally remapped to Trbvh.

Parameters

| | | |
|----|---------------------|--|
| in | <i>acceleration</i> | The acceleration structure handle |
| in | <i>builder</i> | String value specifying the builder type |

Return values

Relevant return values:

- [RT_SUCCESS](#)

- [RT_ERROR_INVALID_VALUE](#)

History

[rtAccelerationSetBuilder](#) was introduced in OptiX 1.0.

See also [rtAccelerationGetBuilder](#), [rtAccelerationSetProperty](#)

3.8.3.12 **RTresult** RTAPI **rtAccelerationSetData** (**RTacceleration** *acceleration*, **const void *** *data*, **RTsize** *size*)

Deprecated in OptiX 4.0. Should not be called.

3.8.3.13 **RTresult** RTAPI **rtAccelerationSetProperty** (**RTacceleration** *acceleration*, **const char *** *name*, **const char *** *value*)

Sets an acceleration structure property.

Description

[rtAccelerationSetProperty](#) sets a named property value for an acceleration structure. Properties can be used to fine tune the way an acceleration structure is built, in order to achieve faster build times or better ray tracing performance. Properties are evaluated and applied by the acceleration structure during build time, and different builders recognize different properties. Setting a property will never fail as long as *acceleration* is a valid handle. Properties that are not recognized by an acceleration structure will be ignored.

The following is a list of the properties used by the individual builders:

- "refit": Available in: Trbv, Bvh If set to "1", the builder will only readjust the node bounds of the bounding volume hierarchy instead of constructing it from scratch. Refit is only effective if there is an initial BVH already in place, and the underlying geometry has undergone relatively modest deformation. In this case, the builder delivers a very fast BVH update without sacrificing too much ray tracing performance. The default is "0".
- "vertex_buffer_name": Available in: Trbv, Sbv The name of the buffer variable holding triangle vertex data. Each vertex consists of 3 floats. The default is "vertex_buffer".
- "vertex_buffer_stride": Available in: Trbv, Sbv The offset between two vertices in the vertex buffer, given in bytes. The default value is "0", which assumes the vertices are tightly packed.
- "index_buffer_name": Available in: Trbv, Sbv The name of the buffer variable holding vertex index data. The entries in this buffer are indices of type int, where each index refers to one entry in the vertex buffer. A sequence of three indices represents one triangle. If no index buffer is given, the vertices in the vertex buffer are assumed to be a list of triangles, i.e. every 3 vertices in a row form a triangle. The default is "index_buffer".
- "index_buffer_stride": Available in: Trbv, Sbv The offset between two indices in the index buffer, given in bytes. The default value is "0", which assumes the indices are tightly packed.
- "chunk_size": Available in: Trbv Number of bytes to be used for a partitioned acceleration structure build. If no chunk size is set, or set to "0", the chunk size is chosen automatically. If set to "-1", the chunk size is unlimited. The minimum chunk size is 64MB. Please note that specifying a small chunk size reduces the peak-memory footprint of the Trbv but can result in slower rendering performance.

Parameters

| | | |
|----|---------------------|---|
| in | <i>acceleration</i> | The acceleration structure handle |
| in | <i>name</i> | String value specifying the name of the property |
| in | <i>value</i> | String value specifying the value of the property |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtAccelerationSetProperty](#) was introduced in OptiX 1.0.

See also [rtAccelerationGetProperty](#), [rtAccelerationSetBuilder](#),

3.8.3.14 RTresult RTAPI rtAccelerationSetTraverser (RTacceleration *acceleration*, const char * *traverser*)

Deprecated in OptiX 4.0. Setting a traverser is no longer necessary and will be ignored.

3.8.3.15 RTresult RTAPI rtAccelerationValidate (RTacceleration *acceleration*)

Validates the state of an acceleration structure.

Description

[rtAccelerationValidate](#) checks *acceleration* for completeness. If *acceleration* is not valid, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|----|---------------------|-----------------------------------|
| in | <i>acceleration</i> | The acceleration structure handle |
|----|---------------------|-----------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtAccelerationValidate](#) was introduced in OptiX 1.0.

See also [rtAccelerationCreate](#)

3.8.3.16 RTresult RTAPI rtBufferBindProgressiveStream (RTbuffer *stream*, RTbuffer *source*)

Bind a stream buffer to an output buffer source.

Description

Binds an output buffer to a progressive stream. The output buffer thereby becomes the data source for the stream. To form a valid output/stream pair, the stream buffer must be of format [RT_FORMAT_UNSIGNED_BYTE4](#), and the output buffer must be of format [RT_FORMAT_FLOAT3](#) or [RT_FORMAT_FLOAT4](#). The use of [RT_FORMAT_FLOAT4](#) is recommended for performance reasons, even if the fourth component is unused. The output buffer must be of type [RT_BUFFER_OUTPUT](#); it may not be of type [RT_BUFFER_INPUT_OUTPUT](#).

Parameters

| | | |
|----|---------------|---|
| in | <i>stream</i> | The stream buffer for which the source is to be specified |
| in | <i>source</i> | The output buffer to function as the stream's source |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtBufferBindProgressiveStream](#) was introduced in OptiX 3.8.

See also [rtBufferCreate](#) [rtBufferSetAttribute](#) [rtBufferGetAttribute](#)

3.8.3.17 RTresult RTAPI rtBufferCreate (RTcontext *context*, unsigned int *bufferdesc*, RTbuffer * *buffer*)

Creates a new buffer object.

Description

[rtBufferCreate](#) allocates and returns a new handle to a new buffer object in **buffer* associated with *context*. The backing storage of the buffer is managed by OptiX. A buffer is specified by a bitwise *or* combination of a *type* and *flags* in *bufferdesc*. The supported types are:

- [RT_BUFFER_INPUT](#)
- [RT_BUFFER_OUTPUT](#)
- [RT_BUFFER_INPUT_OUTPUT](#)
- [RT_BUFFER_PROGRESSIVE_STREAM](#)

The type values are used to specify the direction of data flow from the host to the OptiX devices. [RT_BUFFER_INPUT](#) specifies that the host may only write to the buffer and the device may only read from the buffer. [RT_BUFFER_OUTPUT](#) specifies the opposite, read only access on the host and write only access on the device. Devices and the host may read and write from buffers of type [RT_BUFFER_INPUT_OUTPUT](#). Reading or writing to a buffer of the incorrect type (e.g., the host writing to a buffer of type [RT_BUFFER_OUTPUT](#)) is undefined. [RT_BUFFER_PROGRESSIVE_STREAM](#) is used to receive stream updates generated by progressive launches (see [rtContextLaunchProgressive2D](#)).

The supported flags are:

- [RT_BUFFER_GPU_LOCAL](#)
- [RT_BUFFER_COPY_ON_DIRTY](#)
- [RT_BUFFER_LAYERED](#)
- [RT_BUFFER_CUBEMAP](#)

If [RT_BUFFER_LAYERED](#) flag is set, buffer depth specifies the number of layers, not the depth of a 3D buffer. If [RT_BUFFER_CUBEMAP](#) flag is set, buffer depth specifies the number of cube faces, not the depth of a 3D buffer. See details in [rtBufferSetSize3D](#)

Flags can be used to optimize data transfers between the host and its devices. The flag [RT_BUFFER_GPU_LOCAL](#) can only be used in combination with [RT_BUFFER_INPUT_OUTPUT](#). [RT_BUFFER_INPUT_OUTPUT](#) and [RT_BUFFER_GPU_LOCAL](#) used together specify a buffer that allows the host to *only* write, and the device to read *and* write data. The written data will never be visible on the host side and will generally not be visible on other devices.

If [rtBufferGetDevicePointer](#) has been called for a single device for a given buffer, the user can change the buffer's content on that device through the pointer. OptiX must then synchronize the new buffer contents to all devices. These synchronization copies occur at every [rtContextLaunch](#), unless the buffer is created with [RT_BUFFER_COPY_ON_DIRTY](#). In this case, [rtBufferMarkDirty](#) can be used to notify OptiX that the buffer has been dirtied and must be synchronized.

Returns [RT_ERROR_INVALID_VALUE](#) if *buffer* is *NULL*.

Parameters

| | | |
|-----|-------------------|---|
| in | <i>context</i> | The context to create the buffer in |
| in | <i>bufferdesc</i> | Bitwise <i>or</i> combination of the <i>type</i> and <i>flags</i> of the new buffer |
| out | <i>buffer</i> | The return handle for the buffer object |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferCreate](#) was introduced in OptiX 1.0.

[RT_BUFFER_GPU_LOCAL](#) was introduced in OptiX 2.0.

See also [rtBufferCreateFromGLBO](#), [rtBufferDestroy](#), [rtBufferMarkDirty](#) [rtBufferBindProgressiveStream](#)

3.8.3.18 RTresult RTAPI rtBufferDestroy (RTbuffer *buffer*)

Destroys a buffer object.

Description

[rtBufferDestroy](#) removes *buffer* from its context and deletes it. *buffer* should be a value returned by [rtBufferCreate](#). After the call, *buffer* is no longer a valid handle. Any API object that referenced *buffer* will have its reference invalidated.

Parameters

| | | |
|----|---------------|---------------------------------|
| in | <i>buffer</i> | Handle of the buffer to destroy |
|----|---------------|---------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferDestroy](#) was introduced in OptiX 1.0.

See also [rtBufferCreate](#), [rtBufferCreateFromGLBO](#)

3.8.3.19 RTresult RTAPI rtBufferGetAttribute (RTbuffer *buffer*, RTbufferattribute *attrib*, RTsize *size*, void * *p*)

Query a buffer attribute.

Description

[rtBufferGetAttribute](#) is used to query buffer attributes. For a list of available attributes, please refer to [rtBufferSetAttribute](#).

Parameters

| | | |
|-----|---------------|--|
| in | <i>buffer</i> | The buffer to query the attribute from |
| in | <i>attrib</i> | The attribute to query |
| in | <i>size</i> | The size of the attribute value, in bytes. For string attributes, this is the maximum buffer size the returned string will use (including a terminating null character). |
| out | <i>p</i> | Pointer to the attribute value to be filled in. Must point to valid memory of at least <i>size</i> bytes. |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtBufferGetAttribute](#) was introduced in OptiX 3.8.

See also [rtBufferSetAttribute](#)

3.8.3.20 RTresult RTAPI rtBufferGetContext (RTbuffer *buffer*, RTcontext * *context*)

Returns the context object that created this buffer.

Description

[rtBufferGetContext](#) returns a handle to the context that created *buffer* in **context*. If **context* is *NULL*, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|----------------|--|
| in | <i>buffer</i> | The buffer to be queried for its context |
| out | <i>context</i> | The return handle for the buffer's context |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferGetContext](#) was introduced in OptiX 1.0.

See also [rtContextCreate](#)

3.8.3.21 RTresult RTAPI rtBufferGetDimensionality (RTbuffer *buffer*, unsigned int * *dimensionality*)

Gets the dimensionality of this buffer object.

Description

[rtBufferGetDimensionality](#) returns the dimensionality of *buffer* in **dimensionality*. The value returned will be one of 1, 2 or 3, corresponding to 1D, 2D and 3D buffers, respectively.

Parameters

| | | |
|-----|-----------------------|---|
| in | <i>buffer</i> | The buffer to be queried for its dimensionality |
| out | <i>dimensionality</i> | The return handle for the buffer's dimensionality |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferGetDimensionality](#) was introduced in OptiX 1.0.

See also [rtBufferSetSize{1-2-3}D](#)

3.8.3.22 RTresult RTAPI rtBufferGetElementSize (RTbuffer *buffer*, RTsize * *size_of_element*)

Returns the size of a buffer's individual elements.

Description

[rtBufferGetElementSize](#) queries the size of a buffer's elements. The target buffer is specified by *buffer*, which should be a value returned by [rtBufferCreate](#). The size, in bytes, of the buffer's individual elements is returned in **element_size_return*. Returns [RT_ERROR_INVALID_VALUE](#) if given a *NULL* pointer.

Parameters

| | | |
|-----|------------------------|--|
| in | <i>buffer</i> | Specifies the buffer to be queried |
| out | <i>size_of_element</i> | Returns the size of the buffer's individual elements |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_UNKNOWN](#)

History

[rtBufferGetElementSize](#) was introduced in OptiX 1.0.

See also [rtBufferSetElementSize](#), [rtBufferCreate](#)

3.8.3.23 RTresult RTAPI rtBufferGetFormat (RTbuffer *buffer*, RTformat * *format*)

Gets the format of this buffer.

Description

[rtBufferGetFormat](#) returns, in **format*, the format of *buffer*. See [rtBufferSetFormat](#) for a listing of [RTbuffer](#) values.

Parameters

| | | |
|-----|---------------|---|
| in | <i>buffer</i> | The buffer to be queried for its format |
| out | <i>format</i> | The return handle for the buffer's format |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferGetFormat](#) was introduced in OptiX 1.0.

See also [rtBufferSetFormat](#), [rtBufferGetFormat](#)

3.8.3.24 RTresult RTAPI rtBufferGetId (RTbuffer *buffer*, int * *buffer_id*)

Gets an id suitable for use with buffers of buffers.

Description

[rtBufferGetId](#) returns an ID for the provided buffer. The returned ID is used on the device to reference the buffer. It needs to be copied into a buffer of type [RT_FORMAT_BUFFER_ID](#) or used in a [rtBufferId](#) object.. If **buffer_id* is *NULL* or the *buffer* is not a valid RTbuffer, returns [RT_ERROR_INVALID_VALUE](#). [RT_BUFFER_ID_NULL](#) can be used as a sentinel for a non-existent buffer, since this value will never be returned as a valid buffer id.

Parameters

| | | |
|-----|------------------|-------------------------------------|
| in | <i>buffer</i> | The buffer to be queried for its id |
| out | <i>buffer_id</i> | The returned ID of the buffer |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtBufferGetId](#) was introduced in OptiX 3.5.

See also [rtContextGetBufferFromId](#)

3.8.3.25 RTresult RTAPI rtBufferGetMipLevelCount (RTbuffer *buffer*, unsigned int * *level*)

Gets the number of mipmap levels of this buffer object.

Description

[rtBufferGetMipLevelCount](#) returns the number of mipmap levels. Default number of MIP levels is 1.

Parameters

| | | |
|-----|---------------|--|
| in | <i>buffer</i> | The buffer to be queried for its number of mipmap levels |
| out | <i>level</i> | The return number of mipmap levels |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferGetMipLevelCount](#) was introduced in OptiX 3.9.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.8.3.26 RTresult RTAPI rtBufferGetMipLevelSize1D (RTbuffer *buffer*, unsigned int *level*, RTsize * *width*)

Gets the width of buffer specific MIP level.

Description

[rtBufferGetMipLevelSize1D](#) stores the width of *buffer* in **width*.

Parameters

| | | |
|-----|---------------|---|
| in | <i>buffer</i> | The buffer to be queried for its dimensions |
| in | <i>level</i> | The buffer MIP level index to be queried for its dimensions |
| out | <i>width</i> | The return handle for the buffer's width Return values |

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtBufferGetMipLevelSize1D](#) was introduced in OptiX 3.9.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.8.3.27 **RTresult** RTAPI [rtBufferGetMipLevelSize2D](#) (**RTbuffer** *buffer*, unsigned int *level*, RTsize * *width*, RTsize * *height*)

Gets the width, height of buffer specific MIP level.

Description

[rtBufferGetMipLevelSize2D](#) stores the width, height of *buffer* in **width* and **height* respectively.

Parameters

| | | |
|-----|---------------|--|
| in | <i>buffer</i> | The buffer to be queried for its dimensions |
| in | <i>level</i> | The buffer MIP level index to be queried for its dimensions |
| out | <i>width</i> | The return handle for the buffer's width |
| out | <i>height</i> | The return handle for the buffer's height Return values |

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferGetMipLevelSize2D](#) was introduced in OptiX 3.9.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.8.3.28 **RTresult** RTAPI [rtBufferGetMipLevelSize3D](#) (**RTbuffer** *buffer*, unsigned int *level*, RTsize * *width*, RTsize * *height*, RTsize * *depth*)

Gets the width, height and depth of buffer specific MIP level.

Description

[rtBufferGetMipLevelSize3D](#) stores the width, height and depth of *buffer* in **width*, **height* and **depth*, respectively.

Parameters

| | | |
|-----|---------------|---|
| in | <i>buffer</i> | The buffer to be queried for its dimensions |
| in | <i>level</i> | The buffer MIP level index to be queried for its dimensions |
| out | <i>width</i> | The return handle for the buffer's width |
| out | <i>height</i> | The return handle for the buffer's height |
| out | <i>depth</i> | The return handle for the buffer's depth Return values |

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtBufferGetMipLevelSize3D](#) was introduced in OptiX 3.9.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.8.3.29 **RTresult** RTAPI [rtBufferGetProgressiveUpdateReady](#) (**RTbuffer** *buffer*, int * *ready*, unsigned int * *subframe_count*, unsigned int * *max_subframes*)

Check whether stream buffer content has been updated by a Progressive Launch.

Description

Returns whether or not the result of a progressive launch in *buffer* has been updated since the last time this function was called. A client application should use this call in its main render/display loop to poll for frame refreshes after initiating a progressive launch. If *subframe_count* and *max_subframes* are non-null, they will be filled with the corresponding counters if and only if *ready* returns 1.

Note that this call does not stop a progressive render.

Parameters

| | | |
|-----|-----------------------|---|
| in | <i>buffer</i> | The stream buffer to be queried |
| out | <i>ready</i> | Ready flag. Will be set to 1 if an update is available, or 0 if no update is available. |
| out | <i>subframe_count</i> | The number of subframes accumulated in the latest result |
| out | <i>max_subframes</i> | The <i>max_subframes</i> parameter as specified in the call to rtContextLaunchProgressive2D |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtBufferGetProgressiveUpdateReady](#) was introduced in OptiX 3.8.

See also [rtContextLaunchProgressive2D](#)

3.8.3.30 **RTresult** RTAPI [rtBufferGetSize1D](#) (**RTbuffer** *buffer*, **RTsize** * *width*)

Get the width of this buffer.

Description

[rtBufferGetSize1D](#) stores the width of *buffer* in **width*.

Parameters

| | | |
|-----|---------------|---|
| in | <i>buffer</i> | The buffer to be queried for its dimensions |
| out | <i>width</i> | The return handle for the buffer's width |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferGetSize1D](#) was introduced in OptiX 1.0.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.8.3.31 **RTresult** RTAPI [rtBufferGetSize2D](#) (**RTbuffer** *buffer*, **RTsize** * *width*, **RTsize** * *height*)

Gets the width and height of this buffer.

Description

[rtBufferGetSize2D](#) stores the width and height of *buffer* in **width* and **height*, respectively.

Parameters

| | | |
|-----|---------------|---|
| in | <i>buffer</i> | The buffer to be queried for its dimensions |
| out | <i>width</i> | The return handle for the buffer's width |
| out | <i>height</i> | The return handle for the buffer's height |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferGetSize2D](#) was introduced in OptiX 1.0.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.8.3.32 **RTresult** RTAPI [rtBufferGetSize3D](#) (**RTbuffer** *buffer*, **RTsize** * *width*, **RTsize** * *height*, **RTsize** * *depth*)

Gets the width, height and depth of this buffer.

Description

[rtBufferGetSize3D](#) stores the width, height and depth of *buffer* in **width*, **height* and **depth*, respectively.

Parameters

| | | |
|-----|---------------|---|
| in | <i>buffer</i> | The buffer to be queried for its dimensions |
| out | <i>width</i> | The return handle for the buffer's width |
| out | <i>height</i> | The return handle for the buffer's height |
| out | <i>depth</i> | The return handle for the buffer's depth |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferGetSize3D](#) was introduced in OptiX 1.0.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSizev](#)

3.8.3.33 RTresult RTAPI rtBufferGetSizev (RTbuffer *buffer*, unsigned int *dimensionality*, RTsize * *dims*)

Gets the dimensions of this buffer.

Description

[rtBufferGetSizev](#) stores the dimensions of *buffer* in **dims*. The number of dimensions returned is specified by *dimensionality*. The storage at *dims* must be large enough to hold the number of requested buffer dimensions.

Parameters

| | | |
|-----|-----------------------|---|
| in | <i>buffer</i> | The buffer to be queried for its dimensions |
| in | <i>dimensionality</i> | The number of requested dimensions |
| out | <i>dims</i> | The array of dimensions to store to |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferGetSizev](#) was introduced in OptiX 1.0.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#)

3.8.3.34 RTresult RTAPI rtBufferMap (RTbuffer *buffer*, void ** *user_pointer*)

Maps a buffer object to the host.

Description

[rtBufferMap](#) returns a pointer, accessible by the host, in **user_pointer* that contains a mapped copy of the contents of *buffer*. The memory pointed to by **user_pointer* can be written to or read from, depending on the type of *buffer*. For example, this code snippet demonstrates creating and filling an input buffer with floats.

```
RTbuffer buffer;
float* data;
rtBufferCreate(context, RT_BUFFER_INPUT, &buffer);
rtBufferSetFormat(buffer, RT_FORMAT_FLOAT);
rtBufferSetSizeID(buffer, 10);
rtBufferMap(buffer, (void*)&data);
for(int i = 0; i < 10; ++i)
    data[i] = 4.f * i;
rtBufferUnmap(buffer);
```

If *buffer* has already been mapped, returns [RT_ERROR_ALREADY_MAPPED](#). If *buffer* has size zero, the returned pointer is undefined

Note that this call does not stop a progressive render if called on a stream buffer.

Parameters

| | | |
|-----|---------------------|--|
| in | <i>buffer</i> | The buffer to be mapped |
| out | <i>user_pointer</i> | Return handle to a user pointer where the buffer will be mapped to |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_ALREADY_MAPPED](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferMap](#) was introduced in OptiX 1.0.

See also [rtBufferUnmap](#), [rtBufferMapEx](#), [rtBufferUnmapEx](#)

3.8.3.35 **RTresult** **RTAPI** [rtBufferMapEx](#) ([RTbuffer](#) *buffer*, unsigned int *map_flags*, unsigned int *level*, void * *user_owned*, void ** *optix_owned*)

Maps mipmap level of buffer object to the host.

Description

[rtBufferMapEx](#) makes the buffer contents available on the host, either by returning a pointer in **optix_owned*, or by copying the contents to a memory location pointed to by *user_owned*. Calling [rtBufferMapEx](#) with proper map flags can result in better performance than using [rtBufferMap](#), because fewer synchronization copies are required in certain situations. [rtBufferMapEx](#) with *map_flags* = [RT_BUFFER_MAP_READ_WRITE](#) and *leve* = 0 is equivalent to [rtBufferMap](#).

Note that this call does not stop a progressive render if called on a stream buffer.

Parameters

| | | |
|----|------------------|-------------------------------|
| in | <i>buffer</i> | The buffer to be mapped |
| in | <i>map_flags</i> | Map flags, see below |
| in | <i>level</i> | The mipmap level to be mapped |

| | | |
|------------------|--------------------------|--|
| <code>in</code> | <code>user_owned</code> | Not yet supported. Must be NULL |
| <code>out</code> | <code>optix_owned</code> | Return handle to a user pointer where the buffer will be mapped to |

The following flags are supported for `map_flags`. They are mutually exclusive:

- [RT_BUFFER_MAP_READ](#)
- [RT_BUFFER_MAP_WRITE](#)
- [RT_BUFFER_MAP_READ_WRITE](#)
- [RT_BUFFER_MAP_WRITE_DISCARD](#)

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_ALREADY_MAPPED](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferMapEx](#) was introduced in OptiX 3.9.

See also [rtBufferMap](#), [rtBufferUnmap](#), [rtBufferUnmapEx](#)

3.8.3.36 `RTresult RTAPI rtBufferSetAttribute (RTbuffer buffer, RTbufferattribute attrib, RTsize size, void * p)`

Set a buffer attribute.

Description

Sets a buffer attribute. Currently, all available attributes refer to stream buffers only, and attempting to set them on a non-stream buffer will generate an error.

Each attribute can have a different size. The sizes are given in the following list:

- [RT_BUFFER_ATTRIBUTE_STREAM_FORMAT](#) `strlen(input_string)`
- [RT_BUFFER_ATTRIBUTE_STREAM_BITRATE](#) `sizeof(int)`
- [RT_BUFFER_ATTRIBUTE_STREAM_FPS](#) `sizeof(int)`
- [RT_BUFFER_ATTRIBUTE_STREAM_GAMMA](#) `sizeof(float)`

[RT_BUFFER_ATTRIBUTE_STREAM_FORMAT](#) sets the encoding format used for streams sent over the network, specified as a string. The default is "auto". Various other common stream and image formats are available (e.g. "h264", "png"). This attribute has no effect if the progressive API is used locally.

[RT_BUFFER_ATTRIBUTE_STREAM_BITRATE](#) sets the target bitrate for streams sent over the network, if the stream format supports it. The data is specified as a 32-bit integer. The default is 5000000. This attribute has no effect if the progressive API is used locally or if the stream format does not support variable bitrates.

[RT_BUFFER_ATTRIBUTE_STREAM_FPS](#) sets the target update rate per second for streams sent over the network, if the stream format supports it. The data is specified as a 32-bit integer. The default is 30. This attribute has no effect if the progressive API is used locally or if the stream format does not support variable framerates.

[RT_BUFFER_ATTRIBUTE_STREAM_GAMMA](#) sets the gamma value for the built-in tonemapping operator. The data is specified as a 32-bit float, the default is 1.0. Tonemapping is executed before encoding the accumulated output into the stream, i.e. on the server side if remote rendering is used. See the section on Buffers below for more details.

Parameters

| | | |
|----|---------------|---|
| in | <i>buffer</i> | The buffer on which to set the attribute |
| in | <i>attrib</i> | The attribute to set |
| in | <i>size</i> | The size of the attribute value, in bytes |
| in | <i>p</i> | Pointer to the attribute value |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtBufferSetAttribute](#) was introduced in OptiX 3.8.

See also [rtBufferGetAttribute](#)

3.8.3.37 RTresult RTAPI rtBufferSetElementSize (RTbuffer *buffer*, RTsize *size_of_element*)

Modifies the size in bytes of a buffer's individual elements.

Description

[rtBufferSetElementSize](#) modifies the size in bytes of a buffer's user-formatted elements. The target buffer is specified by *buffer*, which should be a value returned by [rtBufferCreate](#) and should have format [RT_FORMAT_USER](#). The new size of the buffer's individual elements is specified by *element_size* and should not be 0. If the buffer has format [RT_FORMAT_USER](#), and *element_size* is not 0, then the buffer's individual element size is set to *element_size* and all storage associated with the buffer is reset. Otherwise, this call has no effect and returns either [RT_ERROR_TYPE_MISMATCH](#) if the buffer does not have format [RT_FORMAT_USER](#) or [RT_ERROR_INVALID_VALUE](#) if the buffer has format [RT_FORMAT_USER](#) but *element_size* is 0.

Parameters

| | | |
|----|------------------------|---|
| in | <i>buffer</i> | Specifies the buffer to be modified |
| in | <i>size_of_element</i> | Specifies the new size in bytes of the buffer's individual elements |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

[rtBufferSetElementSize](#) was introduced in OptiX 1.0.

See also [rtBufferGetElementSize](#), [rtBufferCreate](#)

3.8.3.38 RTresult RTAPI rtBufferSetFormat (RTbuffer *buffer*, RTformat *format*)

Sets the format of this buffer.

Description

[rtBufferSetFormat](#) changes the *format* of *buffer* to the specified value. The data elements of the buffer will have the specified type and can either be vector formats, or a user-defined type whose size is specified with [rtBufferSetElementSize](#). Possible values for *format* are:

- [RT_FORMAT_HALF](#)

- `RT_FORMAT_HALF2`
- `RT_FORMAT_HALF3`
- `RT_FORMAT_HALF4`
- `RT_FORMAT_FLOAT`
- `RT_FORMAT_FLOAT2`
- `RT_FORMAT_FLOAT3`
- `RT_FORMAT_FLOAT4`
- `RT_FORMAT_BYTE`
- `RT_FORMAT_BYTE2`
- `RT_FORMAT_BYTE3`
- `RT_FORMAT_BYTE4`
- `RT_FORMAT_UNSIGNED_BYTE`
- `RT_FORMAT_UNSIGNED_BYTE2`
- `RT_FORMAT_UNSIGNED_BYTE3`
- `RT_FORMAT_UNSIGNED_BYTE4`
- `RT_FORMAT_SHORT`
- `RT_FORMAT_SHORT2`
- `RT_FORMAT_SHORT3`
- `RT_FORMAT_SHORT4`
- `RT_FORMAT_UNSIGNED_SHORT`
- `RT_FORMAT_UNSIGNED_SHORT2`
- `RT_FORMAT_UNSIGNED_SHORT3`
- `RT_FORMAT_UNSIGNED_SHORT4`
- `RT_FORMAT_INT`
- `RT_FORMAT_INT2`
- `RT_FORMAT_INT3`
- `RT_FORMAT_INT4`
- `RT_FORMAT_UNSIGNED_INT`
- `RT_FORMAT_UNSIGNED_INT2`
- `RT_FORMAT_UNSIGNED_INT3`
- `RT_FORMAT_UNSIGNED_INT4`
- `RT_FORMAT_USER`

Parameters

| | | |
|----|---------------|-----------------------------------|
| in | <i>buffer</i> | The buffer to have its format set |
| in | <i>format</i> | The target format of the buffer |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferSetFormat](#) was introduced in OptiX 1.0.

See also [rtBufferSetFormat](#), [rtBufferGetFormat](#), [rtBufferGetFormat](#), [rtBufferGetElementSize](#), [rtBufferSetElementSize](#)

3.8.3.39 RTresult RTAPI rtBufferSetMipLevelCount (RTbuffer *buffer*, unsigned int *levels*)

Sets the MIP level count of a buffer.

Description

[rtBufferSetMipLevelCount](#) sets the number of MIP levels to *levels*. The default number of MIP levels is 1. Fails with [RT_ERROR_ALREADY_MAPPED](#) if called on a buffer that is mapped.

Parameters

| | | |
|----|---------------|---------------------------------|
| in | <i>buffer</i> | The buffer to be resized |
| in | <i>width</i> | The width of the resized buffer |
| in | <i>levels</i> | Number of mip levels |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_ALREADY_MAPPED](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferSetMipLevelCount](#) was introduced in OptiX 3.9.

See also [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.8.3.40 RTresult RTAPI rtBufferSetSize1D (RTbuffer *buffer*, RTsize *width*)

Sets the width and dimensionality of this buffer.

Description

[rtBufferSetSize1D](#) sets the dimensionality of *buffer* to 1 and sets its width to *width*. Fails with [RT_ERROR_ALREADY_MAPPED](#) if called on a buffer that is mapped.

Parameters

| | | |
|----|---------------|---------------------------------|
| in | <i>buffer</i> | The buffer to be resized |
| in | <i>width</i> | The width of the resized buffer |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_ALREADY_MAPPED](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferSetSize1D](#) was introduced in OptiX 1.0.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.8.3.41 RTresult RTAPI rtBufferSetSize2D (RTbuffer *buffer*, RTsize *width*, RTsize *height*)

Sets the width, height and dimensionality of this buffer.

Description

[rtBufferSetSize2D](#) sets the dimensionality of *buffer* to 2 and sets its width and height to *width* and *height*, respectively. If *width* or *height* is zero, they both must be zero. Fails with [RT_ERROR_ALREADY_MAPPED](#) if called on a buffer that is mapped.

Parameters

| | | |
|----|---------------|----------------------------------|
| in | <i>buffer</i> | The buffer to be resized |
| in | <i>width</i> | The width of the resized buffer |
| in | <i>height</i> | The height of the resized buffer |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_ALREADY_MAPPED](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferSetSize2D](#) was introduced in OptiX 1.0.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize3D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.8.3.42 RTresult RTAPI rtBufferSetSize3D (RTbuffer *buffer*, RTsize *width*, RTsize *height*, RTsize *depth*)

Sets the width, height, depth and dimensionality of a buffer.

Description

[rtBufferSetSize3D](#) sets the dimensionality of *buffer* to 3 and sets its width, height and depth to *width*, *height* and *depth*, respectively. If *width*, *height* or *depth* is zero, they all must be zero.

A 1D layered mipmapped buffer is allocated if *height* is 1 and the [RT_BUFFER_LAYERED](#) flag was set at buffer creating. The number of layers is determined by the *depth*. A 2D layered mipmapped buffer is allocated if the [RT_BUFFER_LAYERED](#) flag was set at buffer creating. The number of layers is determined by the *depth*. A cubemap mipmapped buffer is allocated if the [RT_BUFFER_CUBEMAP](#) flag was set at buffer creating. *width* must be equal to *height* and the number of cube faces is determined by the *depth*, it must be six or a multiple of six, if the [RT_BUFFER_LAYERED](#) flag was also set. Layered, mipmapped and cubemap buffers are supported only as texture buffers.

Fails with [RT_ERROR_ALREADY_MAPPED](#) if called on a buffer that is mapped.

Parameters

| | | |
|----|---------------|----------------------------------|
| in | <i>buffer</i> | The buffer to be resized |
| in | <i>width</i> | The width of the resized buffer |
| in | <i>height</i> | The height of the resized buffer |
| in | <i>depth</i> | The depth of the resized buffer |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_ALREADY_MAPPED](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferSetSize3D](#) was introduced in OptiX 1.0.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSizev](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.8.3.43 RTresult RTAPI rtBufferSetSizev (RTbuffer *buffer*, unsigned int *dimensionality*, const RTsize * *dims*)

Sets the dimensionality and dimensions of a buffer.

Description

[rtBufferSetSizev](#) sets the dimensionality of *buffer* to *dimensionality* and sets the dimensions of the buffer to the values stored at **dims*, which must contain a number of values equal to *dimensionality*. If any of values of *dims* is zero they must all be zero.

Parameters

| | | |
|----|-----------------------|--|
| in | <i>buffer</i> | The buffer to be resized |
| in | <i>dimensionality</i> | The dimensionality the buffer will be resized to |
| in | <i>dims</i> | The array of sizes for the dimension of the resize |

Return values

Relevant return values:

- [RT_SUCCESS](#)

- [RT_ERROR_ALREADY_MAPPED](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferSetSizev](#) was introduced in OptiX 1.0.

See also [rtBufferSetMipLevelCount](#), [rtBufferSetSize1D](#), [rtBufferSetSize2D](#), [rtBufferSetSize3D](#), [rtBufferGetMipLevelSize1D](#), [rtBufferGetMipLevelSize2D](#), [rtBufferGetMipLevelSize3D](#), [rtBufferGetMipLevelCount](#), [rtBufferGetSize1D](#), [rtBufferGetSize2D](#), [rtBufferGetSize3D](#), [rtBufferGetSizev](#)

3.8.3.44 RTresult RTAPI rtBufferUnmap (RTbuffer *buffer*)

Unmaps a buffer's storage from the host.

Description

[rtBufferUnmap](#) unmaps a buffer from the host after a call to [rtBufferMap](#). [rtContextLaunch](#) cannot be called while buffers are still mapped to the host. A call to [rtBufferUnmap](#) that does not follow a matching [rtBufferMap](#) call will return [RT_ERROR_INVALID_VALUE](#).

Note that this call does not stop a progressive render if called with a stream buffer.

Parameters

| | | |
|----|---------------|---------------------|
| in | <i>buffer</i> | The buffer to unmap |
|----|---------------|---------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferUnmap](#) was introduced in OptiX 1.0.

See also [rtBufferMap](#), [rtBufferMapEx](#), [rtBufferUnmapEx](#)

3.8.3.45 RTresult RTAPI rtBufferUnmapEx (RTbuffer *buffer*, unsigned int *level*)

Unmaps mipmap level storage from the host.

Description

[rtBufferUnmapEx](#) unmaps buffer level from the host after a call to [rtBufferMapEx](#). [rtContextLaunch](#) cannot be called while buffers are still mapped to the host. A call to [rtBufferUnmapEx](#) that does not follow a matching [rtBufferMapEx](#) call will return [RT_ERROR_INVALID_VALUE](#). [rtBufferUnmap](#) is equivalent to [rtBufferUnmapEx](#) with *level* = 0.

Note that this call does not stop a progressive render if called with a stream buffer.

Parameters

| | | |
|----|---------------|---------------------|
| in | <i>buffer</i> | The buffer to unmap |
|----|---------------|---------------------|

| | | |
|-----------|--------------|---------------------------|
| <i>in</i> | <i>level</i> | The mipmap level to unmap |
|-----------|--------------|---------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferUnmapEx](#) was introduced in OptiX 3.9.

See also [rtBufferMap](#), [rtBufferUnmap](#), [rtBufferMapEx](#)

3.8.3.46 RTresult RTAPI rtBufferValidate (RTbuffer *buffer*)

Validates the state of a buffer.

Description

[rtBufferValidate](#) checks *buffer* for completeness. If *buffer* has not had its dimensionality, size or format set, this call will return [RT_ERROR_INVALID_CONTEXT](#).

Parameters

| | | |
|-----------|---------------|------------------------|
| <i>in</i> | <i>buffer</i> | The buffer to validate |
|-----------|---------------|------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtBufferValidate](#) was introduced in OptiX 1.0.

See also [rtBufferCreate](#), [rtBufferCreateFromGLBO](#) [rtContextValidate](#)

3.8.3.47 RTresult RTAPI rtContextCompile (RTcontext *context*)

Deprecated in OptiX 4.0. Calling this function has no effect. The kernel is automatically compiled at launch if needed.

3.8.3.48 RTresult RTAPI rtContextCreate (RTcontext * *context*)

Creates a new context object.

Description

[rtContextCreate](#) allocates and returns a handle to a new context object. Returns [RT_ERROR_INVALID_VALUE](#) if passed a *NULL* pointer.

Parameters

| | | |
|-----|----------------|------------------------------------|
| out | <i>context</i> | Handle to context for return value |
|-----|----------------|------------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_NO_DEVICE](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextCreate](#) was introduced in OptiX 1.0.

See also

3.8.3.49 `RTresult RTAPI rtContextDeclareVariable (RTcontext context, const char * name, RTvariable * v)`

Declares a new named variable associated with this context.

Description

[rtContextDeclareVariable](#) - Declares a new variable named *name* and associated with this context. Only a single variable of a given name can exist for a given context and any attempt to create multiple variables with the same name will cause a failure with a return value of [RT_ERROR_VARIABLE_REDECLARED](#). Returns [RT_ERROR_INVALID_VALUE](#) if passed a *NULL* pointer. Return [RT_ERROR_ILLEGAL_SYMBOL](#) if *name* is not syntactically valid.

Parameters

| | | |
|-----|----------------|--|
| in | <i>context</i> | The context node to which the variable will be attached |
| in | <i>name</i> | The name that identifies the variable to be queried |
| out | <i>v</i> | Pointer to variable handle used to return the new object |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_VARIABLE_REDECLARED](#)

History

[rtContextDeclareVariable](#) was introduced in OptiX 1.0.

See also [rtGeometryDeclareVariable](#), [rtGeometryInstanceDeclareVariable](#), [rtMaterialDeclareVariable](#), [rtProgramDeclareVariable](#), [rtSelectorDeclareVariable](#), [rtContextGetVariable](#), [rtContextGetVariableCount](#), [rtContextQueryVariable](#), [rtContextRemoveVariable](#)

3.8.3.50 `RTresult RTAPI rtContextDestroy (RTcontext context)`

Destroys a context and frees all associated resources.

Description

[rtContextDestroy](#) frees all resources, including OptiX objects, associated with this object. Returns [RT_ERROR_INVALID_VALUE](#) if passed a *NULL* context. [RT_ERROR_LAUNCH_FAILED](#) may be returned if a previous call to [rtContextLaunch](#) failed.

Parameters

| | | |
|----|----------------|----------------------------------|
| in | <i>context</i> | Handle of the context to destroy |
|----|----------------|----------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_LAUNCH_FAILED](#)

History

[rtContextDestroy](#) was introduced in OptiX 1.0.

See also [rtContextCreate](#)

3.8.3.51 **RTresult** RTAPI [rtContextGetAttribute](#) (**RTcontext** *context*, **RTcontextattribute** *attrib*, **RTsize** *size*, **void ****p*)

Returns an attribute specific to an OptiX context.

Description

[rtContextGetAttribute](#) returns in *p* the value of the per context attribute specified by *attrib*.

Each attribute can have a different size. The sizes are given in the following list:

- [RT_CONTEXT_ATTRIBUTE_MAX_TEXTURE_COUNT](#) sizeof(int)
- [RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS](#) sizeof(int)
- [RT_CONTEXT_ATTRIBUTE_USED_HOST_MEMORY](#) sizeof(RTsize)
- [RT_CONTEXT_ATTRIBUTE_AVAILABLE_DEVICE_MEMORY](#) sizeof(RTsize)

[RT_CONTEXT_ATTRIBUTE_MAX_TEXTURE_COUNT](#) queries the maximum number of textures handled by OptiX. For OptiX versions below 2.5 this value depends on the number of textures supported by CUDA.

[RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS](#) queries the number of host CPU threads OptiX can use for various tasks.

[RT_CONTEXT_ATTRIBUTE_USED_HOST_MEMORY](#) queries the amount of host memory allocated by OptiX.

[RT_CONTEXT_ATTRIBUTE_AVAILABLE_DEVICE_MEMORY](#) queries the amount of free device memory.

Some attributes are used to get per device information. In contrast to [rtDeviceGetAttribute](#), these attributes are determined by the context and are therefore queried through the context. This is done by adding the attribute with the OptiX device ordinal number when querying the attribute. The following are per device attributes.

[RT_CONTEXT_ATTRIBUTE_AVAILABLE_DEVICE_MEMORY](#)

Parameters

| | | |
|-----|----------------|--|
| in | <i>context</i> | The context object to be queried |
| in | <i>attrib</i> | Attribute to query |
| in | <i>size</i> | Size of the attribute being queried. Parameter <i>p</i> must have at least this much memory allocated |
| out | <i>p</i> | Return pointer where the value of the attribute will be copied into. This must point to at least <i>size</i> bytes of memory |

Return values

Relevant return values:

- [RT_SUCCESS](#)

- [RT_ERROR_INVALID_VALUE](#) - Can be returned if *size* does not match the proper size of the attribute, if *p* is *NULL*, or if *attribute+ordinal* does not correspond to an OptiX device

History

[rtContextGetAttribute](#) was introduced in OptiX 2.0.

See also [rtContextGetDeviceCount](#), [rtContextSetAttribute](#), [rtDeviceGetAttribute](#)

3.8.3.52 RTresult RTAPI rtContextGetBufferFromId (RTcontext *context*, int *buffer_id*, RTbuffer * *buffer*)

Gets an RTbuffer corresponding to the buffer id.

Description

[rtContextGetBufferFromId](#) returns a handle to the buffer in **buffer* corresponding to the *buffer_id* supplied. If *buffer_id* does not map to a valid buffer handle, **buffer* is *NULL* or if *context* is invalid, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|------------------|---|
| in | <i>context</i> | The context the buffer should be originated from |
| in | <i>buffer_id</i> | The ID of the buffer to query |
| out | <i>buffer</i> | The return handle for the buffer object corresponding to the <i>buffer_id</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetBufferFromId](#) was introduced in OptiX 3.5.

See also [rtBufferGetId](#)

3.8.3.53 RTresult RTAPI rtContextGetDeviceCount (RTcontext *context*, unsigned int * *count*)

Query the number of devices currently being used.

Description

[rtContextGetDeviceCount](#) - Query the number of devices currently being used.

Parameters

| | | |
|-----|----------------|---------------------------------------|
| in | <i>context</i> | The context containing the devices |
| out | <i>count</i> | Return parameter for the device count |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetDeviceCount](#) was introduced in OptiX 2.0.

See also [rtContextSetDevices](#), [rtContextGetDevices](#)

3.8.3.54 RTresult RTAPI rtContextGetDevices (RTcontext context, int * devices)

Retrieve a list of hardware devices being used by the kernel.

Description

[rtContextGetDevices](#) retrieves a list of hardware devices used by the context. Note that the device numbers are OptiX device ordinals, which may not be the same as CUDA device ordinals. Use [rtDeviceGetAttribute](#) with [RT_DEVICE_ATTRIBUTE_CUDA_DEVICE_ORDINAL](#) to query the CUDA device corresponding to a particular OptiX device.

Parameters

| | | |
|-----|----------------|--|
| in | <i>context</i> | The context to which the hardware list is applied |
| out | <i>devices</i> | Return parameter for the list of devices. The memory must be able to hold entries numbering least the number of devices as returned by rtContextGetDeviceCount |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetDevices](#) was introduced in OptiX 2.0.

See also [rtContextSetDevices](#), [rtContextGetDeviceCount](#)

3.8.3.55 RTresult RTAPI rtContextGetEntryPointCount (RTcontext context, unsigned int * num_entry_points)

Query the number of entry points for this context.

Description

[rtContextGetEntryPointCount](#) passes back the number of entry points associated with this context in *num_entry_points*. Returns [RT_ERROR_INVALID_VALUE](#) if passed a *NULL* pointer.

Parameters

| | | |
|-----|-------------------------|---|
| in | <i>context</i> | The context node to be queried |
| out | <i>num_entry_points</i> | Return parameter for passing back the entry point count |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetEntryPointCount](#) was introduced in OptiX 1.0.

See also [rtContextSetEntryPointCount](#)

3.8.3.56 void RTAPI rtContextGetErrorString (RTcontext context, RTresult code, const char ** return_string)

Returns the error string associated with a given error.

Description

`rtContextGetErrorString` return a descriptive string given an error code. If *context* is valid and additional information is available from the last OptiX failure, it will be appended to the generic error code description. *return_string* will be set to point to this string. The memory *return_string* points to will be valid until the next API call that returns a string.

Parameters

| | | |
|-----|----------------------|--|
| in | <i>context</i> | The context object to be queried, or <i>NULL</i> |
| in | <i>code</i> | The error code to be converted to string |
| out | <i>return_string</i> | The return parameter for the error string |

Return values

[rtContextGetErrorString](#) does not return a value

History

[rtContextGetErrorString](#) was introduced in OptiX 1.0.

See also

3.8.3.57 `RTresult RTAPI rtContextGetExceptionEnabled (RTcontext context, RException exception, int * enabled)`

Query whether a specified exception is enabled.

Description

[rtContextGetExceptionEnabled](#) passes back *1* in **enabled* if the given exception is enabled, *0* otherwise. *exception* specifies the type of exception to be queried. For a list of available types, see [rtContextSetExceptionEnabled](#). If *exception* is [RT_EXCEPTION_ALL](#), *enabled* is set to *1* only if all possible exceptions are enabled.

Parameters

| | | |
|-----|------------------|--|
| in | <i>context</i> | The context to be queried |
| in | <i>exception</i> | The exception of which to query the state |
| out | <i>enabled</i> | Return parameter to store whether the exception is enabled |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetExceptionEnabled](#) was introduced in OptiX 1.1.

See also [rtContextSetExceptionEnabled](#), [rtContextSetExceptionProgram](#), [rtContextGetExceptionProgram](#), [rtGetExceptionCode](#), [rtThrow](#), [rtPrintExceptionDetails](#)

3.8.3.58 `RTresult RTAPI rtContextGetExceptionProgram (RTcontext context, unsigned int entry_point_index, RTprogram * program)`

Queries the exception program associated with the given context and entry point.

Description

[rtContextGetExceptionProgram](#) passes back the exception program associated with the given context and entry point. This program is set via [rtContextSetExceptionProgram](#). Returns [RT_ERROR_INVALID_VALUE](#) if given an invalid entry point index or *NULL* pointer.

Parameters

| | | |
|----|--------------------------|---|
| in | <i>context</i> | The context node associated with the exception program |
| in | <i>entry_point_index</i> | The entry point index for the desired exception program |

| | | |
|-----|----------------|---|
| out | <i>program</i> | Return parameter to store the exception program |
|-----|----------------|---|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetExceptionProgram](#) was introduced in OptiX 1.0.

See also [rtContextSetExceptionProgram](#), [rtContextSetEntryPointCount](#), [rtContextSetExceptionEnabled](#), [rtContextGetExceptionEnabled](#), [rtGetExceptionCode](#), [rtThrow](#), [rtPrintExceptionDetails](#)

3.8.3.59 **RTresult** RTAPI [rtContextGetMissProgram](#) (**RTcontext** *context*, unsigned int *ray_type_index*, **RTprogram** * *program*)

Queries the miss program associated with the given context and ray type.

Description

[rtContextGetMissProgram](#) passes back the miss program associated with the given context and ray type. This program is set via [rtContextSetMissProgram](#). Returns [RT_ERROR_INVALID_VALUE](#) if given an invalid ray type index or a *NULL* pointer.

Parameters

| | | |
|-----|-----------------------|---|
| in | <i>context</i> | The context node associated with the miss program |
| in | <i>ray_type_index</i> | The ray type index for the desired miss program |
| out | <i>program</i> | Return parameter to store the miss program |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetMissProgram](#) was introduced in OptiX 1.0.

See also [rtContextSetMissProgram](#), [rtContextGetRayTypeCount](#)

3.8.3.60 **RTresult** RTAPI [rtContextGetPrintBufferSize](#) (**RTcontext** *context*, **RTsize** * *buffer_size_bytes*)

Get the current size of the print buffer.

Description

[rtContextGetPrintBufferSize](#) is used to query the buffer size available to hold data generated by [rtPrintf](#). Returns [RT_ERROR_INVALID_VALUE](#) if passed a *NULL* pointer.

Parameters

| | | |
|-----|--------------------------|---|
| in | <i>context</i> | The context from which to query the print buffer size |
| out | <i>buffer_size_bytes</i> | The returned print buffer size in bytes |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetPrintBufferSize](#) was introduced in OptiX 1.0.

See also [rtPrintf](#), [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextGetPrintLaunchIndex](#)

3.8.3.61 RTresult RTAPI rtContextGetPrintEnabled (RTcontext context, int * enabled)

Query whether text printing from programs is enabled.

Description

[rtContextGetPrintEnabled](#) passes back *1* if text printing from programs through [rtPrintf](#) is currently enabled for this context; *0* otherwise. Returns [RT_ERROR_INVALID_VALUE](#) if passed a *NULL* pointer.

Parameters

| | | |
|-----|----------------|---|
| in | <i>context</i> | The context to be queried |
| out | <i>enabled</i> | Return parameter to store whether printing is enabled |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetPrintEnabled](#) was introduced in OptiX 1.0.

See also [rtPrintf](#), [rtContextSetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextGetPrintLaunchIndex](#)

3.8.3.62 RTresult RTAPI rtContextGetPrintLaunchIndex (RTcontext context, int * x, int * y, int * z)

Gets the active print launch index.

Description

[rtContextGetPrintLaunchIndex](#) is used to query for which launch indices [rtPrintf](#) generates output. The initial value of (x,y,z) is (-1,-1,-1), which generates output for all indices.

Parameters

| | | |
|-----|----------------|---|
| in | <i>context</i> | The context from which to query the print launch index |
| out | <i>x</i> | Returns the launch index in the x dimension to which the output of rtPrintf invocations is limited. Will not be written to if a <i>NULL</i> pointer is passed |
| out | <i>y</i> | Returns the launch index in the y dimension to which the output of rtPrintf invocations is limited. Will not be written to if a <i>NULL</i> pointer is passed |
| out | <i>z</i> | Returns the launch index in the z dimension to which the output of rtPrintf invocations is limited. Will not be written to if a <i>NULL</i> pointer is passed |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetPrintLaunchIndex](#) was introduced in OptiX 1.0.

See also [rtPrintf](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#)

3.8.3.63 RTresult RTAPI rtContextGetProgramFromId (RTcontext context, int program_id, RTprogram * program)

Gets an RTprogram corresponding to the program id.

Description

[rtContextGetProgramFromId](#) returns a handle to the program in *program corresponding to the *program_id* supplied. If *program_id* is not a valid program handle, *program is set to *NULL*. Returns [RT_ERROR_INVALID_VALUE](#) if *context* is invalid or *program_id* is not a valid program handle.

Parameters

| | | |
|-----|-------------------|---|
| in | <i>context</i> | The context the program should be originated from |
| in | <i>program_id</i> | The ID of the program to query |
| out | <i>program</i> | The return handle for the program object corresponding to the <i>program_id</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetProgramFromId](#) was introduced in OptiX 3.6.

See also [rtProgramGetId](#)

3.8.3.64 RTresult RTAPI rtContextGetRayGenerationProgram (RTcontext context, unsigned int entry_point_index, RTprogram * program)

Queries the ray generation program associated with the given context and entry point.

Description

[rtContextGetRayGenerationProgram](#) passes back the ray generation program associated with the given context and entry point. This program is set via [rtContextSetRayGenerationProgram](#). Returns [RT_ERROR_INVALID_VALUE](#) if given an invalid entry point index or *NULL* pointer.

Parameters

| | | |
|-----|--------------------------|--|
| in | <i>context</i> | The context node associated with the ray generation program |
| in | <i>entry_point_index</i> | The entry point index for the desired ray generation program |
| out | <i>program</i> | Return parameter to store the ray generation program |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetRayGenerationProgram](#) was introduced in OptiX 1.0.

See also [rtContextSetRayGenerationProgram](#)

3.8.3.65 RTresult RTAPI rtContextGetRayTypeCount (RTcontext context, unsigned int * num_ray_types)

Query the number of ray types associated with this context.

Description

[rtContextGetRayTypeCount](#) passes back the number of entry points associated with this context in *num_ray_types*. Returns [RT_ERROR_INVALID_VALUE](#) if passed a *NULL* pointer.

Parameters

| | | |
|-----|----------------------|---|
| in | <i>context</i> | The context node to be queried |
| out | <i>num_ray_types</i> | Return parameter to store the number of ray types |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetRayTypeCount](#) was introduced in OptiX 1.0.

See also [rtContextSetRayTypeCount](#)

3.8.3.66 RTresult RTAPI rtContextGetRunningState (RTcontext *context*, int * *running*)

Query whether the given context is currently running.

Description

This function is currently unimplemented and it is provided as a placeholder for a future implementation.

Parameters

| | | |
|-----|----------------|---|
| in | <i>context</i> | The context node to be queried |
| out | <i>running</i> | Return parameter to store the running state |

Return values

Since unimplemented, this function will always throw an assertion failure.

History

[rtContextGetRunningState](#) was introduced in OptiX 1.0.

See also [rtContextLaunch1D](#), [rtContextLaunch2D](#), [rtContextLaunch3D](#)

3.8.3.67 RTresult RTAPI rtContextGetStackSize (RTcontext *context*, RTsize * *stack_size_bytes*)

Query the stack size for this context.

Description

[rtContextGetStackSize](#) passes back the stack size associated with this context in *stack_size_bytes*. Returns [RT_ERROR_INVALID_VALUE](#) if passed a *NULL* pointer.

Parameters

| | | |
|-----|-------------------------|---|
| in | <i>context</i> | The context node to be queried |
| out | <i>stack_size_bytes</i> | Return parameter to store the size of the stack |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetStackSize](#) was introduced in OptiX 1.0.

See also [rtContextSetStackSize](#)

3.8.3.68 **RTresult** RTAPI **rtContextGetTextureSamplerFromId** (**RTcontext** *context*, int *sampler_id*, **RTtexturesampler** * *sampler*)

Gets an **RTtexturesampler** corresponding to the texture id.

Description

[rtTextureSamplerGetId](#) returns a handle to the texture sampler in **sampler* corresponding to the *sampler_id* supplied. If *sampler_id* does not map to a valid texture handle, **sampler* is **NULL** or if *context* is invalid, returns **RT_ERROR_INVALID_VALUE**.

Parameters

| | | |
|-----|-------------------|---|
| in | <i>context</i> | The context the sampler should be originated from |
| in | <i>sampler_id</i> | The ID of the sampler to query |
| out | <i>sampler</i> | The return handle for the sampler object corresponding to the <i>sampler_id</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetTextureSamplerFromId](#) was introduced in OptiX 3.5.

See also [rtTextureSamplerGetId](#)

3.8.3.69 **RTresult** RTAPI **rtContextGetVariable** (**RTcontext** *context*, unsigned int *index*, **RTvariable** * *v*)

Queries an indexed variable associated with this context.

Description

[rtContextGetVariable](#) queries the variable at position *index* in the variable array from *context* and stores the result in the parameter *v*. A variable must be declared first with [rtContextDeclareVariable](#) and *index* must be in the range [*0*, [rtContextGetVariableCount](#) - 1].

Parameters

| | | |
|-----|----------------|--|
| in | <i>context</i> | The context node to be queried for an indexed variable |
| in | <i>index</i> | The index that identifies the variable to be queried |
| out | <i>v</i> | Return value to store the queried variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetVariable](#) was introduced in OptiX 1.0.

See also [rtGeometryGetVariable](#), [rtGeometryInstanceGetVariable](#), [rtMaterialGetVariable](#), [rtProgramGetVariable](#), [rtSelectorGetVariable](#), [rtContextDeclareVariable](#), [rtContextGetVariableCount](#), [rtContextQueryVariable](#), [rtContextRemoveVariable](#)

3.8.3.70 **RTresult** RTAPI **rtContextGetVariableCount** (**RTcontext** *context*, unsigned int * *count*)

Returns the number of variables associated with this context.

Description

[rtContextGetVariableCount](#) returns the number of variables that are currently attached to *context*. Returns [RT_ERROR_INVALID_VALUE](#) if passed a *NULL* pointer.

Parameters

| | | |
|-----|----------------|--|
| in | <i>context</i> | The context to be queried for number of attached variables |
| out | <i>count</i> | Return parameter to store the number of variables |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextGetVariableCount](#) was introduced in OptiX 1.0.

See also [rtGeometryGetVariableCount](#), [rtGeometryInstanceGetVariableCount](#), [rtMaterialGetVariableCount](#), [rtProgramGetVariableCount](#), [rtSelectorGetVariable](#), [rtContextDeclareVariable](#), [rtContextGetVariable](#), [rtContextQueryVariable](#), [rtContextRemoveVariable](#)

3.8.3.71 **RTresult** RTAPI [rtContextLaunch1D](#) (**RTcontext** *context*, unsigned int *entry_point_index*, RTsize *image_width*)

Executes the computation kernel for a given context.

Description

[rtContextLaunch](#) functions execute the computation kernel associated with the given context. If the context has not yet been compiled, or if the context has been modified since the last compile, [rtContextLaunch](#) will recompile the kernel internally. Acceleration structures of the context which are marked dirty will be updated and their dirty flags will be cleared. Similarly, validation will occur if necessary. The ray generation program specified by *entry_point_index* will be invoked once for every element (pixel or voxel) of the computation grid specified by *image_width*, *image_height*, and *image_depth*.

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_INVALID_SOURCE](#)
- [RT_ERROR_LAUNCH_FAILED](#)

History

[rtContextLaunch](#) was introduced in OptiX 1.0.

See also [rtContextGetRunningState](#), [rtContextValidate](#)

Parameters

| | | |
|----|--------------------------|-------------------------------------|
| in | <i>context</i> | The context to be executed |
| in | <i>entry_point_index</i> | The initial entry point into kernel |
| in | <i>image_width</i> | Width of the computation grid |

3.8.3.72 **RTresult** RTAPI [rtContextLaunch2D](#) (**RTcontext** *context*, unsigned int *entry_point_index*, RTsize *image_width*, RTsize *image_height*)

Parameters

| | | |
|----|--------------------------|-------------------------------------|
| in | <i>context</i> | The context to be executed |
| in | <i>entry_point_index</i> | The initial entry point into kernel |
| in | <i>image_width</i> | Width of the computation grid |
| in | <i>image_height</i> | Height of the computation grid |

3.8.3.73 `RTresult RTAPI rtContextLaunch3D (RTcontext context, unsigned int entry_point_index, RTsize image_width, RTsize image_height, RTsize image_depth)`

Parameters

| | | |
|----|--------------------------|-------------------------------------|
| in | <i>context</i> | The context to be executed |
| in | <i>entry_point_index</i> | The initial entry point into kernel |
| in | <i>image_width</i> | Width of the computation grid |
| in | <i>image_height</i> | Height of the computation grid |
| in | <i>image_depth</i> | Depth of the computation grid |

3.8.3.74 `RTresult RTAPI rtContextLaunchProgressive2D (RTcontext context, unsigned int entry_index, RTsize width, RTsize height, unsigned int max_subframes)`

Executes a Progressive Launch for a given context.

Description

Starts the (potentially parallel) generation of subframes for progressive rendering. If *max_subframes* is zero, there is no limit on the number of subframes generated. The generated subframes are automatically composited into a single result and streamed to the client at regular intervals, where they can be read by mapping an associated stream buffer. An application can therefore initiate a progressive launch, and then repeatedly map and display the contents of the stream buffer in order to visualize the progressive refinement of the image.

The call is nonblocking. A polling approach should be used to decide when to map and display the stream buffer contents (see [rtBufferGetProgressiveUpdateReady](#)). If a progressive launch is already in progress at the time of the call and its parameters match the initial launch, the call has no effect. Otherwise, the accumulated result will be reset and a new progressive launch will be started.

If any other OptiX function is called while a progressive launch is in progress, it will cause the launch to stop generating new subframes (however, subframes that have already been generated and are currently in flight may still arrive at the client). The only exceptions to this rule are the operations to map a stream buffer, issuing another progressive launch with unchanged parameters, and polling for an update. Those exceptions do not cause the progressive launch to stop generating subframes.

There is no guarantee that the call actually produces any subframes, especially if [rtContextLaunchProgressive2D](#) and other OptiX commands are called in short succession. For example, during an animation, `rtVariableSet` calls may be tightly interleaved with progressive launches, and when rendering remotely the server may decide to skip some of the launches in order to avoid a large backlog in the command pipeline.

Parameters

| | | |
|----|----------------------|---|
| in | <i>context</i> | The context in which the launch is to be executed |
| in | <i>entry_index</i> | The initial entry point into kernel |
| in | <i>width</i> | Width of the computation grid |
| in | <i>height</i> | Height of the computation grid |
| in | <i>max_subframes</i> | The maximum number of subframes to be generated. Set to zero to generate an unlimited number of subframes |

Return values

Relevant return values:

- [RT_SUCCESS](#)

- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_LAUNCH_FAILED](#)

History

[rtContextLaunchProgressive2D](#) was introduced in OptiX 3.8.

See also [rtContextStopProgressive](#) [rtBufferGetProgressiveUpdateReady](#)

3.8.3.75 **RTresult** RTAPI [rtContextQueryVariable](#) (**RTcontext** *context*, **const char ****name*, **RTvariable ****v*)

Returns a named variable associated with this context.

Description

[rtContextQueryVariable](#) queries a variable identified by the string *name* from *context* and stores the result in **v*. A variable must be declared with [rtContextDeclareVariable](#) before it can be queried, otherwise **v* will be set to *NULL*. [RT_ERROR_INVALID_VALUE](#) will be returned if *name* or *v* is *NULL*.

Parameters

| | | |
|-----|----------------|---|
| in | <i>context</i> | The context node to query a variable from |
| in | <i>name</i> | The name that identifies the variable to be queried |
| out | <i>v</i> | Return value to store the queried variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextQueryVariable](#) was introduced in OptiX 1.0.

See also [rtGeometryQueryVariable](#), [rtGeometryInstanceQueryVariable](#), [rtMaterialQueryVariable](#), [rtProgramQueryVariable](#), [rtSelectorQueryVariable](#), [rtContextDeclareVariable](#), [rtContextGetVariableCount](#), [rtContextGetVariable](#), [rtContextRemoveVariable](#)

3.8.3.76 **RTresult** RTAPI [rtContextRemoveVariable](#) (**RTcontext** *context*, **RTvariable** *v*)

Removes a variable from the given context.

Description

[rtContextRemoveVariable](#) removes variable *v* from *context* if present. Returns [RT_ERROR_VARIABLE_NOT_FOUND](#) if the variable is not attached to this context. Returns [RT_ERROR_INVALID_VALUE](#) if passed an invalid variable.

Parameters

| | | |
|----|----------------|--|
| in | <i>context</i> | The context node from which to remove a variable |
| in | <i>v</i> | The variable to be removed |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_VARIABLE_NOT_FOUND](#)

History

[rtContextRemoveVariable](#) was introduced in OptiX 1.0.

See also [rtGeometryRemoveVariable](#), [rtGeometryInstanceRemoveVariable](#), [rtMaterialRemoveVariable](#), [rtProgramRemoveVariable](#), [rtSelectorRemoveVariable](#), [rtContextDeclareVariable](#), [rtContextGetVariable](#), [rtContextGetVariableCount](#), [rtContextQueryVariable](#),

3.8.3.77 **RTresult** RTAPI [rtContextSetAttribute](#) (**RTcontext** *context*, **RTcontextattribute** *attrib*, **RTsize** *size*, **void *** *p*)

Set an attribute specific to an OptiX context.

Description

[rtContextSetAttribute](#) sets *p* as the value of the per context attribute specified by *attrib*.

Each attribute can have a different size. The sizes are given in the following list:

- [RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS](#) sizeof(int)

[RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS](#) sets the number of host CPU threads OptiX can use for various tasks.

Parameters

| | | |
|----|----------------|--|
| in | <i>context</i> | The context object to be modified |
| in | <i>attrib</i> | Attribute to set |
| in | <i>size</i> | Size of the attribute being set |
| in | <i>p</i> | Pointer to where the value of the attribute will be copied from. This must point to at least <i>size</i> bytes of memory |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#) - Can be returned if *size* does not match the proper size of the attribute, or if *p* is *NULL*

History

[rtContextSetAttribute](#) was introduced in OptiX 2.5.

See also [rtContextGetAttribute](#)

3.8.3.78 **RTresult** RTAPI [rtContextSetDevices](#) (**RTcontext** *context*, **unsigned int** *count*, **const int *** *devices*)

Specify a list of hardware devices to be used by the kernel.

Description

[rtContextSetDevices](#) specifies a list of hardware devices to be used during execution of the subsequent trace kernels. Note that the device numbers are OptiX device ordinals, which may not be the same as CUDA device ordinals. Use [rtDeviceGetAttribute](#) with [RT_DEVICE_ATTRIBUTE_CUDA_DEVICE_ORDINAL](#) to query the CUDA device corresponding to a particular OptiX device.

Parameters

| | | |
|----|----------------|---|
| in | <i>context</i> | The context to which the hardware list is applied |
| in | <i>count</i> | The number of devices in the list |
| in | <i>devices</i> | The list of devices |

Return values

Relevant return values:

- [RT_SUCCESS](#)

- [RT_ERROR_NO_DEVICE](#)
- [RT_ERROR_INVALID_DEVICE](#)

History

[rtContextSetDevices](#) was introduced in OptiX 1.0.

See also [rtContextGetDevices](#), [rtContextGetDeviceCount](#)

3.8.3.79 RTresult RTAPI rtContextSetEntryPointCount (RTcontext context, unsigned int num_entry_points)

Set the number of entry points for a given context.

Description

[rtContextSetEntryPointCount](#) sets the number of entry points associated with the given context to *num_entry_points*.

Parameters

| | | |
|----|-------------------------|-----------------------------------|
| in | <i>context</i> | The context to be modified |
| in | <i>num_entry_points</i> | The number of entry points to use |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextSetEntryPointCount](#) was introduced in OptiX 1.0.

See also [rtContextGetEntryPointCount](#)

3.8.3.80 RTresult RTAPI rtContextSetExceptionEnabled (RTcontext context, RTexception exception, int enabled)

Enable or disable an exception.

Description

[rtContextSetExceptionEnabled](#) is used to enable or disable specific exceptions. If an exception is enabled, the exception condition is checked for at runtime, and the exception program is invoked if the condition is met. The exception program can query the type of the caught exception by calling [rtGetExceptionCode](#). *exception* may take one of the following values:

- [RT_EXCEPTION_TEXTURE_ID_INVALID](#)
- [RT_EXCEPTION_BUFFER_ID_INVALID](#)
- [RT_EXCEPTION_INDEX_OUT_OF_BOUNDS](#)
- [RT_EXCEPTION_STACK_OVERFLOW](#)
- [RT_EXCEPTION_BUFFER_INDEX_OUT_OF_BOUNDS](#)
- [RT_EXCEPTION_INVALID_RAY](#)
- [RT_EXCEPTION_INTERNAL_ERROR](#)
- [RT_EXCEPTION_USER](#)
- [RT_EXCEPTION_ALL](#)

[RT_EXCEPTION_TEXTURE_ID_INVALID](#) verifies that every access of a texture id is valid, including use of [RT_TEXTURE_ID_NULL](#) and IDs out of bounds.

[RT_EXCEPTION_BUFFER_ID_INVALID](#) verifies that every access of a buffer id is valid, including use of [RT_BUFFER_ID_NULL](#) and IDs out of bounds.

[RT_EXCEPTION_INDEX_OUT_OF_BOUNDS](#) checks that [rtIntersectChild](#) and [rtReportIntersection](#) are called with a valid index.

[RT_EXCEPTION_STACK_OVERFLOW](#) checks the runtime stack against overflow. The most common cause for an overflow is a too deep [rtTrace](#) recursion tree.

[RT_EXCEPTION_BUFFER_INDEX_OUT_OF_BOUNDS](#) checks every read and write access to [rtBuffer](#) objects to be within valid bounds.

[RT_EXCEPTION_INVALID_RAY](#) checks the each ray's origin and direction values against *NaNs* and *infinity* values.

[RT_EXCEPTION_INTERNAL_ERROR](#) indicates an unexpected internal error in the runtime.

[RT_EXCEPTION_USER](#) is used to enable or disable all user-defined exceptions. The reserved range of exception codes for user-defined exceptions starts at [RT_EXCEPTION_USER](#) (*0x400*) and ends at *0xFFFF*. See [rtThrow](#) for more information.

[RT_EXCEPTION_ALL](#) is a placeholder value which can be used to enable or disable all possible exceptions with a single call to [rtContextSetExceptionEnabled](#).

By default, [RT_EXCEPTION_STACK_OVERFLOW](#) is enabled and all other exceptions are disabled.

Parameters

| | | |
|----|------------------|--|
| in | <i>context</i> | The context for which the exception is to be enabled or disabled |
| in | <i>exception</i> | The exception which is to be enabled or disabled |
| in | <i>enabled</i> | Nonzero to enable the exception, <i>0</i> to disable the exception |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextSetExceptionEnabled](#) was introduced in OptiX 1.1.

See also [rtContextGetExceptionEnabled](#), [rtContextSetExceptionProgram](#), [rtContextGetExceptionProgram](#), [rtGetExceptionCode](#), [rtThrow](#), [rtPrintExceptionDetails](#)

3.8.3.81 **RTresult** RTAPI [rtContextSetExceptionProgram](#) (**RTcontext** *context*, unsigned int *entry_point_index*, **RTprogram** *program*)

Specifies the exception program for a given context entry point.

Description

[rtContextSetExceptionProgram](#) sets *context*'s exception program at entry point *entry_point_index*. [RT_ERROR_INVALID_VALUE](#) is returned if *entry_point_index* is outside of the range [*0*, [rtContextGetEntryPointCount](#) - *1*].

Parameters

| | | |
|----|--------------------------|---|
| in | <i>context</i> | The context node to which the exception program will be added |
| in | <i>entry_point_index</i> | The entry point the program will be associated with |

| | | |
|-----------|----------------|-----------------------|
| <i>in</i> | <i>program</i> | The exception program |
|-----------|----------------|-----------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

[rtContextSetExceptionProgram](#) was introduced in OptiX 1.0.

See also [rtContextGetEntryPointCount](#), [rtContextGetExceptionProgram](#) [rtContextSetExceptionEnabled](#), [rtContextGetExceptionEnabled](#), [rtGetExceptionCode](#), [rtThrow](#), [rtPrintExceptionDetails](#)

3.8.3.82 **RTresult** RTAPI [rtContextSetMissProgram](#) (**RTcontext** *context*, unsigned int *ray_type_index*, **RTprogram** *program*)

Specifies the miss program for a given context ray type.

Description

[rtContextSetMissProgram](#) sets *context*'s miss program associated with ray type *ray_type_index*. [RT_ERROR_INVALID_VALUE](#) is returned if *ray_type_index* is outside of the range $[0, \text{rtContextGetRayTypeCount} - 1]$.

Parameters

| | | |
|-----------|-----------------------|--|
| <i>in</i> | <i>context</i> | The context node to which the miss program will be added |
| <i>in</i> | <i>ray_type_index</i> | The ray type the program will be associated with |
| <i>in</i> | <i>program</i> | The miss program |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

[rtContextSetMissProgram](#) was introduced in OptiX 1.0.

See also [rtContextGetRayTypeCount](#), [rtContextGetMissProgram](#)

3.8.3.83 **RTresult** RTAPI [rtContextSetPrintBufferSize](#) (**RTcontext** *context*, **RTsize** *buffer_size_bytes*)

Set the size of the print buffer.

Description

[rtContextSetPrintBufferSize](#) is used to set the buffer size available to hold data generated by [rtPrintf](#). Returns [RT_ERROR_INVALID_VALUE](#) if it is called after the first invocation of [rtContextLaunch](#).

Parameters

| | | |
|----|---------------------------------|--|
| in | <i>context</i> | The context for which to set the print buffer size |
| in | <i>buffer_size_ - bytes</i> | The print buffer size in bytes |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextSetPrintBufferSize](#) was introduced in OptiX 1.0.

See also [rtPrintf](#), [rtContextSetPrintEnabled](#), [rtContextGetPrintEnabled](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextGetPrintLaunchIndex](#)

3.8.3.84 RTresult RTAPI rtContextSetPrintEnabled (RTcontext *context*, int *enabled*)

Enable or disable text printing from programs.

Description

[rtContextSetPrintEnabled](#) is used to control whether text printing in programs through [rtPrintf](#) is currently enabled for this context.

Parameters

| | | |
|----|----------------|---|
| in | <i>context</i> | The context for which printing is to be enabled or disabled |
| in | <i>enabled</i> | Setting this parameter to a nonzero value enables printing, 0 disables printing |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextSetPrintEnabled](#) was introduced in OptiX 1.0.

See also [rtPrintf](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#), [rtContextGetPrintLaunchIndex](#)

3.8.3.85 RTresult RTAPI rtContextSetPrintLaunchIndex (RTcontext *context*, int *x*, int *y*, int *z*)

Sets the active launch index to limit text output.

Description

[rtContextSetPrintLaunchIndex](#) is used to control for which launch indices [rtPrintf](#) generates output. The initial value of (x,y,z) is (-1,-1,-1), which generates output for all indices.

Parameters

| | | |
|----|----------------|---|
| in | <i>context</i> | The context for which to set the print launch index |
| in | <i>x</i> | The launch index in the x dimension to which to limit the output of rtPrintf invocations. If set to -1, output is generated for all launch indices in the x dimension |

| | | |
|----|---|---|
| in | y | The launch index in the y dimension to which to limit the output of rtPrintf invocations. If set to -1, output is generated for all launch indices in the y dimension |
| in | z | The launch index in the z dimension to which to limit the output of rtPrintf invocations. If set to -1, output is generated for all launch indices in the z dimension |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextSetPrintLaunchIndex](#) was introduced in OptiX 1.0.

See also [rtPrintf](#), [rtContextGetPrintEnabled](#), [rtContextSetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextGetPrintLaunchIndex](#)

3.8.3.86 **RTresult** RTAPI [rtContextSetRayGenerationProgram](#) (**RTcontext** *context*, unsigned int *entry_point_index*, **RTprogram** *program*)

Specifies the ray generation program for a given context entry point.

Description

[rtContextSetRayGenerationProgram](#) sets *context*'s ray generation program at entry point *entry_point_index*. [RT_ERROR_INVALID_VALUE](#) is returned if *entry_point_index* is outside of the range $[0, \text{rtContextGetEntryPointCount} - 1]$.

Parameters

| | | |
|----|--------------------------|---|
| in | <i>context</i> | The context node to which the exception program will be added |
| in | <i>entry_point_index</i> | The entry point the program will be associated with |
| in | <i>program</i> | The ray generation program |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

[rtContextSetRayGenerationProgram](#) was introduced in OptiX 1.0.

See also [rtContextGetEntryPointCount](#), [rtContextGetRayGenerationProgram](#)

3.8.3.87 **RTresult** RTAPI [rtContextSetRayTypeCount](#) (**RTcontext** *context*, unsigned int *num_ray_types*)

Sets the number of ray types for a given context.

Description

[rtContextSetRayTypeCount](#) Sets the number of ray types associated with the given context.

Parameters

| | | |
|----|----------------------|------------------------------------|
| in | <i>context</i> | The context node |
| in | <i>num_ray_types</i> | The number of ray types to be used |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextSetRayTypeCount](#) was introduced in OptiX 1.0.

See also [rtContextGetRayTypeCount](#)

3.8.3.88 RTresult RTAPI rtContextSetRemoteDevice (RTcontext *context*, RTremotedevice *remote_dev*)

Enable rendering on a remote device.

Description

Associates a context with a remote device. If successful, any further OptiX calls will be directed to the remote device and executed there. The context must be an empty, newly created context. In other words, in order to use a context remotely, the call to [rtContextSetRemoteDevice](#) should immediately follow the call to [rtContextCreate](#).

Note that a context that was used for remote rendering cannot be re-used for local rendering by changing devices. However, the Progressive API (that is, [rtContextLaunchProgressive2D](#), stream buffers, etc.) can be used locally by simply not creating a remote device and not calling [rtContextSetRemoteDevice](#).

Only a single remote device can be associated with a context. Switching between different remote devices is not supported.

Parameters

| | | |
|----|-------------------|--|
| in | <i>context</i> | Newly created context to use on the remote device |
| in | <i>remote_dev</i> | Remote device on which rendering is to be executed |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextSetRemoteDevice](#) was introduced in OptiX 3.8.

See also [rtRemoteDeviceCreate](#) [rtRemoteDeviceGetAttribute](#) [rtRemoteDeviceReserve](#) [rtContextLaunchProgressive2D](#)

3.8.3.89 RTresult RTAPI rtContextSetStackSize (RTcontext *context*, RTsize *stack_size_bytes*)

Set the stack size for a given context.

Description

[rtContextSetStackSize](#) sets the stack size for the given context to *stack_size_bytes* bytes. Returns [RT_ERROR_INVALID_VALUE](#) if context is not valid.

Parameters

| | | |
|----|-------------------------|---------------------------------|
| in | <i>context</i> | The context node to be modified |
| in | <i>stack_size_bytes</i> | The desired stack size in bytes |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextSetStackSize](#) was introduced in OptiX 1.0.

See also [rtContextGetStackSize](#)

3.8.3.90 **RTresult** RTAPI [rtContextSetTimeoutCallback](#) (**RTcontext** *context*, **RTtimeoutcallback** *callback*, double *min_polling_seconds*)

Side timeout callback function.

Description

[rtContextSetTimeoutCallback](#) sets an application-side callback function *callback* and a time interval *min_polling_seconds* in seconds. Potentially long-running OptiX API calls such as [rtContextLaunch](#) call the callback function about every *min_polling_seconds* seconds. The core purpose of a timeout callback function is to give the application a chance to do whatever it might need to do frequently, such as handling GUI events.

If the callback function returns true, the API call tries to abort, leaving the context in a clean but unfinished state. Output buffers are left in an unpredictable state. In case an OptiX API call is terminated by a callback function, it returns [RT_TIMEOUT_CALLBACK](#).

As a side effect, timeout functions also help control the OptiX kernel run-time. This can in some cases prevent OptiX kernel launches from running so long that they cause driver timeouts. For example, if *min_polling_seconds* is 0.5 seconds then once the kernel has been running for 0.5 seconds it won't start any new launch indices (calls to a ray generation program). Thus, if the driver's timeout is 2 seconds (the default on Windows), then a launch index may take up to 1.5 seconds without triggering a driver timeout.

[RTtimeoutcallback](#) is defined as *int* (*RTtimeoutcallback)(void).

To unregister a callback function, *callback* needs to be set to *NULL* and *min_polling_seconds* to 0.

Only one timeout callback function can be specified at any time.

Returns [RT_ERROR_INVALID_VALUE](#) if *context* is not valid, if *min_polling_seconds* is negative, if *callback* is *NULL* but *min_polling_seconds* is not 0, or if *callback* is not *NULL* but *min_polling_seconds* is 0.

Parameters

| | | |
|----|----------------------------|---|
| in | <i>context</i> | The context node to be modified |
| in | <i>callback</i> | The function to be called |
| in | <i>min_polling_seconds</i> | The timeout interval after which the function is called |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtContextSetTimeoutCallback](#) was introduced in OptiX 2.5.

See also [rtContextLaunch](#)

3.8.3.91 RTresult RTAPI rtContextStopProgressive (RTcontext context)

Stops a Progressive Launch.

Description

If a progressive launch is currently in progress, calling [rtContextStopProgressive](#) terminates it. Otherwise, the call has no effect. If a launch is stopped using this function, no further subframes will arrive at the client, even if they have already been generated by the server and are currently in flight.

This call should only be used if the application must guarantee that frames generated by previous progressive launches won't be accessed. Do not call [rtContextStopProgressive](#) in the main rendering loop if the goal is only to change OptiX state (e.g. rtVariable values). The call is unnecessary in that case and will degrade performance.

Parameters

| | | |
|-----------|----------------|--|
| <i>in</i> | <i>context</i> | The context associated with the progressive launch |
|-----------|----------------|--|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_INVALID_CONTEXT](#)

History

[rtContextStopProgressive](#) was introduced in OptiX 3.8.

See also [rtContextLaunchProgressive2D](#)

3.8.3.92 RTresult RTAPI rtContextValidate (RTcontext context)

Checks the given context for valid internal state.

Description

[rtContextValidate](#) checks the the given context and all of its associated OptiX objects for a valid state. These checks include tests for presence of necessary programs (e.g. an intersection program for a geometry node), invalid internal state such as *NULL* children in graph nodes, and presence of variables required by all specified programs. [rtContextGetErrorString](#) can be used to retrieve a description of a validation failure.

Parameters

| | | |
|-----------|----------------|-----------------------------|
| <i>in</i> | <i>context</i> | The context to be validated |
|-----------|----------------|-----------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_INVALID_SOURCE](#)

History

[rtContextValidate](#) was introduced in OptiX 1.0.

See also [rtContextGetErrorString](#)

3.8.3.93 `RTresult RTAPI rtDeviceGetAttribute (int ordinal, RTdeviceattribute attrib, RTsize size, void * p)`

Returns an attribute specific to an OptiX device.

Description

`rtDeviceGetAttribute` returns in *p* the value of the per device attribute specified by *attrib* for device *ordinal*.

Each attribute can have a different size. The sizes are given in the following list:

- `RT_DEVICE_ATTRIBUTE_MAX_THREADS_PER_BLOCK` `sizeof(int)`
- `RT_DEVICE_ATTRIBUTE_CLOCK_RATE` `sizeof(int)`
- `RT_DEVICE_ATTRIBUTE_MULTIPROCESSOR_COUNT` `sizeof(int)`
- `RT_DEVICE_ATTRIBUTE_EXECUTION_TIMEOUT_ENABLED` `sizeof(int)`
- `RT_DEVICE_ATTRIBUTE_MAX_HARDWARE_TEXTURE_COUNT` `sizeof(int)`
- `RT_DEVICE_ATTRIBUTE_NAME` up to `size-1`
- `RT_DEVICE_ATTRIBUTE_COMPUTE_CAPABILITY` `sizeof(int2)`
- `RT_DEVICE_ATTRIBUTE_TOTAL_MEMORY` `sizeof(RTsize)`
- `RT_DEVICE_ATTRIBUTE_TCC_DRIVER` `sizeof(int)`
- `RT_DEVICE_ATTRIBUTE_CUDA_DEVICE_ORDINAL` `sizeof(int)`

Parameters

| | | |
|-----|----------------|--|
| in | <i>ordinal</i> | OptiX device ordinal |
| in | <i>attrib</i> | Attribute to query |
| in | <i>size</i> | Size of the attribute being queried. Parameter <i>p</i> must have at least this much memory allocated |
| out | <i>p</i> | Return pointer where the value of the attribute will be copied into. This must point to at least <i>size</i> bytes of memory |

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE` - Can be returned if *size* does not match the proper size of the attribute, if *p* is `NULL`, or if *ordinal* does not correspond to an OptiX device

History

`rtDeviceGetAttribute` was introduced in OptiX 2.0. `RT_DEVICE_ATTRIBUTE_TCC_DRIVER` was introduced in OptiX 3.0. `RT_DEVICE_ATTRIBUTE_CUDA_DEVICE_ORDINAL` was introduced in OptiX 3.0.

See also `rtDeviceGetDeviceCount`, `rtContextGetAttribute`

3.8.3.94 `RTresult RTAPI rtDeviceGetDeviceCount (unsigned int * count)`

Returns the number of OptiX capable devices.

Description

`rtDeviceGetDeviceCount` returns in *count* the number of compute devices that are available in the host system and will be used by OptiX.

Parameters

| | | |
|-----|-------|------------------------------------|
| out | count | Number devices available for OptiX |
|-----|-------|------------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtDeviceGetDeviceCount](#) was introduced in OptiX 1.0.

See also [rtGetVersion](#)

3.8.3.95 RTresult RTAPI rtGeometryCreate (RTcontext context, RTgeometry * geometry)

Creates a new geometry node.

Description

[rtGeometryCreate](#) creates a new geometry node within a context. *context* specifies the target context, and should be a value returned by [rtContextCreate](#). Sets **geometry* to the handle of a newly created geometry within *context*. Returns [RT_ERROR_INVALID_VALUE](#) if *geometry* is *NULL*.

Parameters

| | | |
|-----|----------|--|
| in | context | Specifies the rendering context of the Geometry node |
| out | geometry | New Geometry node handle |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryCreate](#) was introduced in OptiX 1.0.

See also [rtGeometryDestroy](#), [rtGeometrySetBoundingBoxProgram](#), [rtGeometrySetIntersectionProgram](#)

3.8.3.96 RTresult RTAPI rtGeometryDeclareVariable (RTgeometry geometry, const char * name, RTvariable * v)

Declares a new named variable associated with a geometry instance.

Description

[rtGeometryDeclareVariable](#) declares a new variable associated with a geometry node. *geometry* specifies the target geometry node, and should be a value returned by [rtGeometryCreate](#). *name* specifies the name of the variable, and should be a *NULL-terminated* string. If there is currently no variable associated with *geometry* named *name*, a new variable named *name* will be created and associated with *geometry*. Returns the handle of the newly-created variable in **v* or *NULL* otherwise. After declaration, the variable can be queried with [rtGeometryQueryVariable](#) or [rtGeometryGetVariable](#). A declared variable does not have a type until its value is set with one of the [rtVariableSet](#) functions. Once a variable is set, its type cannot be changed anymore.

Parameters

| | | |
|-----|-----------------|---|
| in | <i>geometry</i> | Specifies the associated Geometry node |
| in | <i>name</i> | The name that identifies the variable |
| out | <i>v</i> | Returns a handle to a newly declared variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_REDECLARED](#)
- [RT_ERROR_ILLEGAL_SYMBOL](#)

History

[rtGeometryDeclareVariable](#) was introduced in OptiX 1.0.

See also [Variables](#), [rtGeometryQueryVariable](#), [rtGeometryGetVariable](#), [rtGeometryRemoveVariable](#)

3.8.3.97 RTresult RTAPI rtGeometryDestroy (RTgeometry *geometry*)

Destroys a geometry node.

Description

[rtGeometryDestroy](#) removes *geometry* from its context and deletes it. *geometry* should be a value returned by [rtGeometryCreate](#). Associated variables declared via [rtGeometryDeclareVariable](#) are destroyed, but no child graph nodes are destroyed. After the call, *geometry* is no longer a valid handle.

Parameters

| | | |
|----|-----------------|--|
| in | <i>geometry</i> | Handle of the geometry node to destroy |
|----|-----------------|--|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryDestroy](#) was introduced in OptiX 1.0.

See also [rtGeometryCreate](#), [rtGeometrySetPrimitiveCount](#), [rtGeometryGetPrimitiveCount](#)

3.8.3.98 RTresult RTAPI rtGeometryGetBoundingBoxProgram (RTgeometry *geometry*, RTprogram * *program*)

Returns the attached bounding box program.

Description

[rtGeometryGetBoundingBoxProgram](#) returns the handle *program* for the attached bounding box program of *geometry*.

Parameters

| | | |
|-----|-----------------|--|
| in | <i>geometry</i> | Geometry node handle from which to query program |
| out | <i>program</i> | Handle to attached bounding box program |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGetBoundingBoxProgram](#) was introduced in OptiX 1.0.

See also [rtGeometrySetBoundingBoxProgram](#)

3.8.3.99 **RTresult** RTAPI [rtGeometryGetContext](#) (**RTgeometry** *geometry*, **RTcontext** * *context*)

Returns the context associated with a geometry node.

Description

[rtGeometryGetContext](#) queries a geometry node for its associated context. *geometry* specifies the geometry node to query, and should be a value returned by [rtGeometryCreate](#). Sets **context* to the context associated with *geometry*.

Parameters

| | | |
|-----|-----------------|---|
| in | <i>geometry</i> | Specifies the geometry to query |
| out | <i>context</i> | The context associated with <i>geometry</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGetContext](#) was introduced in OptiX 1.0.

See also [rtGeometryCreate](#)

3.8.3.100 **RTresult** RTAPI [rtGeometryGetIntersectionProgram](#) (**RTgeometry** *geometry*, **RTprogram** * *program*)

Returns the attached intersection program.

Description

[rtGeometryGetIntersectionProgram](#) returns in *program* a handle of the attached intersection program.

Parameters

| | | |
|-----|-----------------|---|
| in | <i>geometry</i> | Geometry node handle to query program |
| out | <i>program</i> | Handle to attached intersection program |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGetIntersectionProgram](#) was introduced in OptiX 1.0.

See also [rtGeometrySetIntersectionProgram](#), [rtProgramCreateFromPTXFile](#), [rtProgramCreateFromPTXString](#)

3.8.3.101 RTresult RTAPI rtGeometryGetPrimitiveCount (RTgeometry *geometry*, unsigned int * *num_primitives*)

Returns the number of primitives.

Description

[rtGeometryGetPrimitiveCount](#) returns for *geometry* the number of set primitives. The number of primitives can be set with [rtGeometrySetPrimitiveCount](#).

Parameters

| | | |
|-----|-----------------------|--|
| in | <i>geometry</i> | Geometry node to query from the number of primitives |
| out | <i>num_primitives</i> | Number of primitives |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGetPrimitiveCount](#) was introduced in OptiX 1.0.

See also [rtGeometrySetPrimitiveCount](#)

3.8.3.102 RTresult RTAPI rtGeometryGetPrimitiveIndexOffset (RTgeometry *geometry*, unsigned int * *index_offset*)

Returns the current primitive index offset.

Description

[rtGeometryGetPrimitiveIndexOffset](#) returns for *geometry* the primitive index offset. The primitive index offset can be set with [rtGeometrySetPrimitiveIndexOffset](#).

Parameters

| | | |
|----|-----------------|---|
| in | <i>geometry</i> | Geometry node to query for the primitive index offset |
|----|-----------------|---|

| | | |
|-----|---------------------|------------------------|
| out | <i>index_offset</i> | Primitive index offset |
|-----|---------------------|------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGeometryGetPrimitiveIndexOffset](#) was introduced in OptiX 3.5.

See also [rtGeometrySetPrimitiveIndexOffset](#)

3.8.3.103 **RTresult** RTAPI [rtGeometryGetVariable](#) (**RTgeometry** *geometry*, unsigned int *index*, **RTvariable** * *v*)

Returns a handle to an indexed variable of a geometry node.

Description

[rtGeometryGetVariable](#) queries the handle of a geometry node's indexed variable. *geometry* specifies the target geometry and should be a value returned by [rtGeometryCreate](#). *index* specifies the index of the variable, and should be a value less than [rtGeometryGetVariableCount](#). If *index* is the index of a variable attached to *geometry*, returns its handle in **v* or *NULL* otherwise. **v* must be declared first with [rtGeometryDeclareVariable](#) before it can be queried.

Parameters

| | | |
|-----|-----------------|--|
| in | <i>geometry</i> | The geometry node from which to query a variable |
| in | <i>index</i> | The index that identifies the variable to be queried |
| out | <i>v</i> | Returns handle to indexed variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_NOT_FOUND](#)

History

[rtGeometryGetVariable](#) was introduced in OptiX 1.0.

See also [rtGeometryDeclareVariable](#), [rtGeometryGetVariableCount](#), [rtGeometryRemoveVariable](#), [rtGeometry-QueryVariable](#)

3.8.3.104 **RTresult** RTAPI [rtGeometryGetVariableCount](#) (**RTgeometry** *geometry*, unsigned int * *count*)

Returns the number of attached variables.

Description

[rtGeometryGetVariableCount](#) queries the number of variables attached to a geometry node. *geometry* specifies the geometry node, and should be a value returned by [rtGeometryCreate](#). After the call, the number of variables attached to *geometry* is returned to **count*.

Parameters

| | | |
|-----|-----------------|--|
| in | <i>geometry</i> | The Geometry node to query from the number of attached variables |
| out | <i>count</i> | Returns the number of attached variables |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGetVariableCount](#) was introduced in OptiX 1.0.

See also [rtGeometryGetVariableCount](#), [rtGeometryDeclareVariable](#), [rtGeometryRemoveVariable](#)

3.8.3.105 **RTresult** RTAPI [rtGeometryGroupCreate](#) (**RTcontext** *context*, **RTgeometrygroup** * *geometrygroup*)

Creates a new geometry group.

Description

[rtGeometryGroupCreate](#) creates a new geometry group within a context. *context* specifies the target context, and should be a value returned by [rtContextCreate](#). Sets **geometrygroup* to the handle of a newly created geometry group within *context*. Returns [RT_ERROR_INVALID_VALUE](#) if *geometrygroup* is *NULL*.

Parameters

| | | |
|-----|----------------------|---|
| in | <i>context</i> | Specifies a context within which to create a new geometry group |
| out | <i>geometrygroup</i> | Returns a newly created geometry group |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGroupCreate](#) was introduced in OptiX 1.0.

See also [rtGeometryGroupDestroy](#), [rtContextCreate](#)

3.8.3.106 **RTresult** RTAPI [rtGeometryGroupDestroy](#) (**RTgeometrygroup** *geometrygroup*)

Destroys a geometry group node.

Description

[rtGeometryGroupDestroy](#) removes *geometrygroup* from its context and deletes it. *geometrygroup* should be a value returned by [rtGeometryGroupCreate](#). No child graph nodes are destroyed. After the call, *geometrygroup* is no longer a valid handle.

Parameters

| | | |
|----|----------------------|--|
| in | <i>geometrygroup</i> | Handle of the geometry group node to destroy |
|----|----------------------|--|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGroupDestroy](#) was introduced in OptiX 1.0.

See also [rtGeometryGroupCreate](#)

3.8.3.107 **RTresult** RTAPI [rtGeometryGroupGetAcceleration](#) (**RTgeometrygroup** *geometrygroup*, **RTacceleration** * *acceleration*)

Returns the acceleration structure attached to a geometry group.

Description

[rtGeometryGroupGetAcceleration](#) returns the acceleration structure attached to a geometry group using [rtGeometryGroupSetAcceleration](#). If no acceleration structure has previously been set, **acceleration* is set to *NULL*.

Parameters

| | | |
|-----|----------------------|--|
| in | <i>geometrygroup</i> | The geometry group handle |
| out | <i>acceleration</i> | The returned acceleration structure object |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGroupGetAcceleration](#) was introduced in OptiX 1.0.

See also [rtGeometryGroupSetAcceleration](#), [rtAccelerationCreate](#)

3.8.3.108 **RTresult** RTAPI [rtGeometryGroupGetChild](#) (**RTgeometrygroup** *geometrygroup*, unsigned int *index*, **RTgeometryinstance** * *geometryinstance*)

Returns a child node of a geometry group.

Description

[rtGeometryGroupGetChild](#) returns the child geometry instance at slot *index* of the parent *geometrygroup*. If no child has been assigned to the given slot, **geometryinstance* is set to *NULL*. Returns [RT_ERROR_INVALID_VALUE](#) if given an invalid child index or *NULL* pointer.

Parameters

| | | |
|-----|-------------------------|--------------------------------------|
| in | <i>geometrygroup</i> | The parent geometry group handle |
| in | <i>index</i> | The index of the child slot to query |
| out | <i>geometryinstance</i> | The returned child geometry instance |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGroupGetChild](#) was introduced in OptiX 1.0.

See also [rtGeometryGroupSetChild](#), [rtGeometryGroupSetChildCount](#), [rtGeometryGroupGetChildCount](#),

3.8.3.109 **RTresult** RTAPI [rtGeometryGroupGetChildCount](#) (**RTgeometrygroup** *geometrygroup*, unsigned int * *count*)

Returns the number of child slots for a group.

Description

[rtGeometryGroupGetChildCount](#) returns the number of child slots allocated using [rtGeometryGroupSetChildCount](#). This includes empty slots which may not yet have actual children assigned by [rtGeometryGroupSetChild](#).

Parameters

| | | |
|-----|----------------------|----------------------------------|
| in | <i>geometrygroup</i> | The parent geometry group handle |
| out | <i>count</i> | Returned number of child slots |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGroupGetChildCount](#) was introduced in OptiX 1.0.

See also [rtGeometryGroupSetChild](#), [rtGeometryGroupGetChild](#), [rtGeometryGroupSetChildCount](#)

3.8.3.110 **RTresult** RTAPI [rtGeometryGroupGetContext](#) (**RTgeometrygroup** *geometrygroup*, **RTcontext** * *context*)

Returns the context associated with a geometry group.

Description

[rtGeometryGroupGetContext](#) queries a geometry group for its associated context. *geometrygroup* specifies the geometry group to query, and must be a value returned by [rtGeometryGroupCreate](#). Sets **context* to the context associated with *geometrygroup*.

Parameters

| | | |
|-----|----------------------|--|
| in | <i>geometrygroup</i> | Specifies the geometry group to query |
| out | <i>context</i> | Returns the context associated with the geometry group |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGroupGetContext](#) was introduced in OptiX 1.0.

See also [rtContextCreate](#), [rtGeometryGroupCreate](#)

3.8.3.111 **RTresult** RTAPI [rtGeometryGroupSetAcceleration](#) (**RTgeometrygroup** *geometrygroup*, **RTacceleration** *acceleration*)

Set the acceleration structure for a group.

Description

[rtGeometryGroupSetAcceleration](#) attaches an acceleration structure to a geometry group. The acceleration structure must have been previously created using [rtAccelerationCreate](#). Every geometry group is required to have an acceleration structure assigned in order to pass validation. The acceleration structure will be built over the primitives contained in all children of the geometry group. This enables a single acceleration structure to be built over primitives of multiple geometry instances. Note that it is legal to attach a single **RTacceleration** object to multiple geometry groups, as long as the underlying geometry of all children is the same. This corresponds to attaching an acceleration structure to multiple groups at higher graph levels using [rtGroupSetAcceleration](#).

Parameters

| | | |
|----|----------------------|--|
| in | <i>geometrygroup</i> | The geometry group handle |
| in | <i>acceleration</i> | The acceleration structure to attach to the geometry group |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGroupSetAcceleration](#) was introduced in OptiX 1.0.

See also [rtGeometryGroupGetAcceleration](#), [rtAccelerationCreate](#), [rtGroupSetAcceleration](#)

3.8.3.112 **RTresult** RTAPI [rtGeometryGroupSetChild](#) (**RTgeometrygroup** *geometrygroup*, unsigned int *index*, **RTgeometryinstance** *geometryinstance*)

Attaches a child node to a geometry group.

Description

[rtGeometryGroupSetChild](#) attaches a new child node *geometryinstance* to the parent node *geometrygroup*. *index* specifies the number of the slot where the child node gets attached. The index value must be lower than the number previously set by [rtGeometryGroupSetChildCount](#).

Parameters

| | | |
|----|-------------------------|--|
| in | <i>geometrygroup</i> | The parent geometry group handle |
| in | <i>index</i> | The index in the parent's child slot array |
| in | <i>geometryinstance</i> | The child node to be attached |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGroupSetChild](#) was introduced in OptiX 1.0.

See also [rtGeometryGroupSetChildCount](#), [rtGeometryGroupGetChildCount](#), [rtGeometryGroupGetChild](#)

3.8.3.113 **RTresult** RTAPI [rtGeometryGroupSetChildCount](#) (**RTgeometrygroup** *geometrygroup*, unsigned int *count*)

Sets the number of child nodes to be attached to the group.

Description

[rtGeometryGroupSetChildCount](#) specifies the number of child slots in this geometry group. Potentially existing links to children at indices greater than *count-1* are removed. If the call increases the number of slots, the newly created slots are empty and need to be filled using [rtGeometryGroupSetChild](#) before validation.

Parameters

| | | |
|----|----------------------|--|
| in | <i>geometrygroup</i> | The parent geometry group handle |
| in | <i>count</i> | Number of child slots to allocate for the geometry group |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGroupSetChildCount](#) was introduced in OptiX 1.0.

See also [rtGeometryGroupGetChild](#), [rtGeometryGroupGetChildCount](#), [rtGeometryGroupSetChild](#)

3.8.3.114 **RTresult** RTAPI [rtGeometryGroupValidate](#) (**RTgeometrygroup** *geometrygroup*)

Validates the state of the geometry group.

Description

[rtGeometryGroupValidate](#) checks *geometrygroup* for completeness. If *geometrygroup* or any of the objects attached to *geometrygroup* are not valid, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|----|----------------------|--|
| in | <i>geometrygroup</i> | Specifies the geometry group to be validated |
|----|----------------------|--|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryGroupValidate](#) was introduced in OptiX 1.0.

See also [rtGeometryGroupCreate](#)

3.8.3.115 **RTresult** RTAPI [rtGeometryInstanceCreate](#) (**RTcontext** *context*, **RTgeometryinstance** * *geometryinstance*)

Creates a new geometry instance node.

Description

[rtGeometryInstanceCreate](#) creates a new geometry instance node within a context. *context* specifies the target context, and should be a value returned by [rtContextCreate](#). Sets **geometryinstance* to the handle of a newly created geometry instance within *context*. Returns [RT_ERROR_INVALID_VALUE](#) if *geometryinstance* is *NULL*.

Parameters

| | | |
|-----|-------------------------|--|
| in | <i>context</i> | Specifies the rendering context of the GeometryInstance node |
| out | <i>geometryinstance</i> | New GeometryInstance node handle |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceCreate](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceDestroy](#), [rtGeometryInstanceDestroy](#), [rtGeometryInstanceGetContext](#)

3.8.3.116 **RTresult** RTAPI [rtGeometryInstanceDeclareVariable](#) (**RTgeometryinstance** *geometryinstance*, **const char** * *name*, **RTvariable** * *v*)

Declares a new named variable associated with a geometry node.

Description

[rtGeometryInstanceDeclareVariable](#) declares a new variable associated with a geometry instance node. *geometryinstance* specifies the target geometry node, and should be a value returned by [rtGeometryInstanceCreate](#). *name* specifies the name of the variable, and should be a *NULL-terminated* string. If there is currently no variable associated with *geometryinstance* named *name*, a new variable named *name* will be created and associated with *geometryinstance*. After the call, **v* will be set to the handle of the newly-created variable. Otherwise, **v* will be

set to `NULL`. After declaration, the variable can be queried with [rtGeometryInstanceQueryVariable](#) or [rtGeometryInstanceGetVariable](#). A declared variable does not have a type until its value is set with one of the `rtVariableSet` functions. Once a variable is set, its type cannot be changed anymore.

Parameters

| | | |
|-----|-------------------------|--|
| in | <i>geometryinstance</i> | Specifies the associated GeometryInstance node |
| in | <i>name</i> | The name that identifies the variable |
| out | <i>v</i> | Returns a handle to a newly declared variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceDeclareVariable](#) was introduced in OptiX 1.0.

See also Variables, [rtGeometryInstanceQueryVariable](#), [rtGeometryInstanceGetVariable](#), [rtGeometryInstanceRemoveVariable](#)

3.8.3.117 RTresult RTAPI rtGeometryInstanceDestroy (RTgeometryinstance *geometryinstance*)

Destroys a geometry instance node.

Description

[rtGeometryInstanceDestroy](#) removes *geometryinstance* from its context and deletes it. *geometryinstance* should be a value returned by [rtGeometryInstanceCreate](#). Associated variables declared via [rtGeometryInstanceDeclareVariable](#) are destroyed, but no child graph nodes are destroyed. After the call, *geometryinstance* is no longer a valid handle.

Parameters

| | | |
|----|-------------------------|---|
| in | <i>geometryinstance</i> | Handle of the geometry instance node to destroy |
|----|-------------------------|---|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceDestroy](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceCreate](#)

3.8.3.118 RTresult RTAPI rtGeometryInstanceGetContext (RTgeometryinstance *geometryinstance*, RTcontext * *context*)

Returns the context associated with a geometry instance node.

Description

[rtGeometryInstanceGetContext](#) queries a geometry instance node for its associated context. *geometryinstance* specifies the geometry node to query, and should be a value returned by [rtGeometryInstanceCreate](#). Sets **context* to the context associated with *geometryinstance*.

Parameters

| | | |
|-----|-------------------------|---------------------------------|
| in | <i>geometryinstance</i> | Specifies the geometry instance |
| out | <i>context</i> | Handle for queried context |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceGetContext](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceGetContext](#)

3.8.3.119 **RTresult** RTAPI [rtGeometryInstanceGetGeometry](#) (**RTgeometryinstance** *geometryinstance*, **RTgeometry** * *geometry*)

Returns the attached Geometry node.

Description

[rtGeometryInstanceGetGeometry](#) sets *geometry* to the handle of the attached Geometry node. If no Geometry node is attached, [RT_ERROR_INVALID_VALUE](#) is returned, else [RT_SUCCESS](#).

Parameters

| | | |
|-----|-------------------------|--|
| in | <i>geometryinstance</i> | GeometryInstance node handle to query geometry |
| out | <i>geometry</i> | Handle to attached Geometry node |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceGetGeometry](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceCreate](#), [rtGeometryInstanceDestroy](#), [rtGeometryInstanceValidate](#), [rtGeometryInstanceSetGeometry](#)

3.8.3.120 **RTresult** RTAPI [rtGeometryInstanceGetMaterial](#) (**RTgeometryinstance** *geometryinstance*, unsigned int *index*, **RTmaterial** * *material*)

Returns a material handle.

Description

[rtGeometryInstanceGetMaterial](#) returns handle *material* for the Material node at position *index* in the material list of *geometryinstance*. Returns [RT_ERROR_INVALID_VALUE](#) if *index* is invalid.

Parameters

| | | |
|-----|-------------------------|--|
| in | <i>geometryinstance</i> | GeometryInstance node handle to query material |
| in | <i>index</i> | Index of material |
| out | <i>material</i> | Handle to material |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceGetMaterial](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceGetMaterialCount](#), [rtGeometryInstanceSetMaterial](#)

3.8.3.121 **RTresult** RTAPI [rtGeometryInstanceGetMaterialCount](#) (**RTgeometryinstance** *geometryinstance*, unsigned int * *count*)

Returns the number of attached materials.

Description

[rtGeometryInstanceGetMaterialCount](#) returns for *geometryinstance* the number of attached Material nodes *count*. The number of materies can be set with [rtGeometryInstanceSetMaterialCount](#).

Parameters

| | | |
|-----|-------------------------|---|
| in | <i>geometryinstance</i> | GeometryInstance node to query from the number of materials |
| out | <i>count</i> | Number of attached materials |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGeometryInstanceGetMaterialCount](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceSetMaterialCount](#)

3.8.3.122 **RTresult** RTAPI [rtGeometryInstanceGetVariable](#) (**RTgeometryinstance** *geometryinstance*, unsigned int *index*, **RTvariable** * *v*)

Returns a handle to an indexed variable of a geometry instance node.

Description

[rtGeometryInstanceGetVariable](#) queries the handle of a geometry instance's indexed variable. *geometryinstance* specifies the target geometry instance and should be a value returned by [rtGeometryInstanceCreate](#). *index* specifies the index of the variable, and should be a value less than [rtGeometryInstanceGetVariableCount](#). If *index* is the index of a variable attached to *geometryinstance*, returns a handle to that variable in **v*, and *NULL* otherwise. **v* must be declared first with [rtGeometryInstanceDeclareVariable](#) before it can be queried.

Parameters

| | | |
|-----|-------------------------|--|
| in | <i>geometryinstance</i> | The GeometryInstance node from which to query a variable |
| in | <i>index</i> | The index that identifies the variable to be queried |
| out | <i>v</i> | Returns handle to indexed variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_NOT_FOUND](#)

History

[rtGeometryInstanceGetVariable](#) was introduced in OptiX 1.0.

See also [rtGeometryDeclareVariable](#), [rtGeometryGetVariableCount](#), [rtGeometryRemoveVariable](#), [rtGeometryQueryVariable](#)

3.8.3.123 **RTresult** RTAPI [rtGeometryInstanceGetVariableCount](#) (**RTgeometryinstance** *geometryinstance*, unsigned int * *count*)

Returns the number of attached variables.

Description

[rtGeometryInstanceGetVariableCount](#) queries the number of variables attached to a geometry instance. *geometryinstance* specifies the geometry instance, and should be a value returned by [rtGeometryInstanceCreate](#). After the call, the number of variables attached to *geometryinstance* is returned to **count*.

Parameters

| | | |
|-----|-------------------------|--|
| in | <i>geometryinstance</i> | The GeometryInstance node to query from the number of attached variables |
| out | <i>count</i> | Returns the number of attached variables |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceGetVariableCount](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceGetVariableCount](#), [rtGeometryInstanceDeclareVariable](#), [rtGeometryInstanceRemoveVariable](#)

3.8.3.124 **RTresult** RTAPI [rtGeometryInstanceQueryVariable](#) (**RTgeometryinstance** *geometryinstance*, const char * *name*, **RTvariable** * *v*)

Returns a handle to a named variable of a geometry node.

Description

[rtGeometryInstanceQueryVariable](#) queries the handle of a geometry instance node's named variable. *geometryinstance* specifies the target geometry instance node, as returned by [rtGeometryInstanceCreate](#). *name* specifies the name of the variable, and should be a *NULL-terminated* string. If *name* is the name of a variable attached to *geometryinstance*, returns a handle to that variable in *v, otherwise *NULL*. Geometry instance variables have to be declared with [rtGeometryInstanceDeclareVariable](#) before they can be queried.

Parameters

| | | |
|-----|-------------------------|---|
| in | <i>geometryinstance</i> | The GeometryInstance node to query from a variable |
| in | <i>name</i> | The name that identifies the variable to be queried |
| out | <i>v</i> | Returns the named variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceQueryVariable](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceDeclareVariable](#), [rtGeometryInstanceRemoveVariable](#), [rtGeometryInstanceGetVariableCount](#), [rtGeometryInstanceGetVariable](#)

3.8.3.125 **RTresult** RTAPI [rtGeometryInstanceRemoveVariable](#) (**RTgeometryinstance** *geometryinstance*, **RTvariable** *v*)

Removes a named variable from a geometry instance node.

Description

[rtGeometryInstanceRemoveVariable](#) removes a named variable from a geometry instance. The target geometry instance is specified by *geometryinstance*, which should be a value returned by [rtGeometryInstanceCreate](#). The variable to be removed is specified by *v*, which should be a value returned by [rtGeometryInstanceDeclareVariable](#). Once a variable has been removed from this geometry instance, another variable with the same name as the removed variable may be declared.

Parameters

| | | |
|----|-------------------------|---|
| in | <i>geometryinstance</i> | The GeometryInstance node from which to remove a variable |
| in | <i>v</i> | The variable to be removed |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_NOT_FOUND](#)

History

[rtGeometryInstanceRemoveVariable](#) was introduced in OptiX 1.0.

See also [rtContextRemoveVariable](#), [rtGeometryInstanceDeclareVariable](#)

3.8.3.126 RTresult RTAPI rtGeometryInstanceSetGeometry (RTgeometryinstance *geometryinstance*, RTgeometry *geometry*)

Attaches a Geometry node.

Description

[rtGeometryInstanceSetGeometry](#) attaches a Geometry node to a GeometryInstance. Only *one* Geometry node can be attached to a GeometryInstance. However, it is at any time possible to attach a different Geometry node.

Parameters

| | | |
|----|-------------------------|--|
| in | <i>geometryinstance</i> | GeometryInstance node handle to attach geometry |
| in | <i>geometry</i> | Geometry handle to attach to <i>geometryinstance</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceSetGeometry](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceGetGeometry](#)

3.8.3.127 RTresult RTAPI rtGeometryInstanceSetMaterial (RTgeometryinstance *geometryinstance*, unsigned int *index*, RTmaterial *material*)

Sets a material.

Description

[rtGeometryInstanceSetMaterial](#) attaches *material* to *geometryinstance* at position *index* in its internal Material node list. *index* must be in the range 0 to [rtGeometryInstanceGetMaterialCount](#) - 1.

Parameters

| | | |
|----|-------------------------|--|
| in | <i>geometryinstance</i> | GeometryInstance node for which to set a material |
| in | <i>index</i> | Index into the material list |
| in | <i>material</i> | Material handle to attach to <i>geometryinstance</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceSetMaterial](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceGetMaterialCount](#), [rtGeometryInstanceSetMaterialCount](#)

3.8.3.128 **RTresult** RTAPI **rtGeometryInstanceSetMaterialCount** (**RTgeometryinstance** *geometryinstance*, unsigned int *count*)

Sets the number of materials.

Description

[rtGeometryInstanceSetMaterialCount](#) sets the number of materials *count* that will be attached to *geometryinstance*. The number of attached materials can be changed at any time. Increasing the number of materials will not modify already assigned materials. Decreasing the number of materials will not modify the remaining already assigned materials.

Parameters

| | | |
|----|-------------------------|--|
| in | <i>geometryinstance</i> | GeometryInstance node to set number of materials |
| in | <i>count</i> | Number of materials to be set |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceSetMaterialCount](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceGetMaterialCount](#)

3.8.3.129 **RTresult** RTAPI **rtGeometryInstanceValidate** (**RTgeometryinstance** *geometryinstance*)

Checks a GeometryInstance node for internal consistency.

Description

[rtGeometryInstanceValidate](#) checks *geometryinstance* for completeness. If *geometryinstance* or any of the objects attached to *geometry* are not valid, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|----|-------------------------|---|
| in | <i>geometryinstance</i> | GeometryInstance node of a model sub-tree to be validated |
|----|-------------------------|---|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryInstanceValidate](#) was introduced in OptiX 1.0.

See also [rtGeometryInstanceCreate](#)

3.8.3.130 **RTresult** RTAPI **rtGeometryIsDirty** (**RTgeometry** *geometry*, int * *dirty*)

Deprecated in OptiX 4.0. Calling this function has no effect.

3.8.3.131 RTresult RTAPI rtGeometryMarkDirty (RTgeometry *geometry*)

Deprecated in OptiX 4.0. Calling this function has no effect.

3.8.3.132 RTresult RTAPI rtGeometryQueryVariable (RTgeometry *geometry*, const char * *name*, RTvariable * *v*)

Returns a handle to a named variable of a geometry node.

Description

[rtGeometryQueryVariable](#) queries the handle of a geometry node's named variable. *geometry* specifies the target geometry node and should be a value returned by [rtGeometryCreate](#). *name* specifies the name of the variable, and should be a *NULL-terminated* string. If *name* is the name of a variable attached to *geometry*, returns a handle to that variable in **v* or *NULL* otherwise. Geometry variables must be declared with [rtGeometryDeclareVariable](#) before they can be queried.

Parameters

| | | |
|-----|-----------------|---|
| in | <i>geometry</i> | The geometry node to query from a variable |
| in | <i>name</i> | The name that identifies the variable to be queried |
| out | <i>v</i> | Returns the named variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_NOT_FOUND](#)

History

[rtGeometryQueryVariable](#) was introduced in OptiX 1.0.

See also [rtGeometryDeclareVariable](#), [rtGeometryRemoveVariable](#), [rtGeometryGetVariableCount](#), [rtGeometryGetVariable](#)

3.8.3.133 RTresult RTAPI rtGeometryRemoveVariable (RTgeometry *geometry*, RTvariable *v*)

Removes a named variable from a geometry node.

Description

[rtGeometryRemoveVariable](#) removes a named variable from a geometry node. The target geometry is specified by *geometry*, which should be a value returned by [rtGeometryCreate](#). The variable to remove is specified by *v*, which should be a value returned by [rtGeometryDeclareVariable](#). Once a variable has been removed from this geometry node, another variable with the same name as the removed variable may be declared.

Parameters

| | | |
|----|-----------------|---|
| in | <i>geometry</i> | The geometry node from which to remove a variable |
| in | <i>v</i> | The variable to be removed |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_NOT_FOUND](#)

History

[rtGeometryRemoveVariable](#) was introduced in OptiX 1.0.

See also [rtContextRemoveVariable](#)

3.8.3.134 RTresult RTAPI rtGeometrySetBoundingBoxProgram (RTgeometry *geometry*, RTprogram *program*)

Sets the bounding box program.

Description

[rtGeometrySetBoundingBoxProgram](#) sets for *geometry* the *program* that computes an axis aligned bounding box for each attached primitive to *geometry*. RTprogram's can be either generated with [rtProgramCreateFromPTXFile](#) or [rtProgramCreateFromPTXString](#). A bounding box program is mandatory for every geometry node.

Parameters

| | | |
|----|-----------------|---|
| in | <i>geometry</i> | The geometry node for which to set the bounding box program |
| in | <i>program</i> | Handle to the bounding box program |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

[rtGeometrySetBoundingBoxProgram](#) was introduced in OptiX 1.0.

See also [rtGeometryGetBoundingBoxProgram](#), [rtProgramCreateFromPTXFile](#), [rtProgramCreateFromPTXString](#)

3.8.3.135 RTresult RTAPI rtGeometrySetIntersectionProgram (RTgeometry *geometry*, RTprogram *program*)

Sets the intersection program.

Description

[rtGeometrySetIntersectionProgram](#) sets for *geometry* the *program* that performs ray primitive intersections. RTprogram's can be either generated with [rtProgramCreateFromPTXFile](#) or [rtProgramCreateFromPTXString](#). An intersection program is mandatory for every geometry node.

Parameters

| | | |
|----|-----------------|---|
| in | <i>geometry</i> | The geometry node for which to set the intersection program |
| in | <i>program</i> | A handle to the ray primitive intersection program |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

[rtGeometrySetIntersectionProgram](#) was introduced in OptiX 1.0.

See also [rtGeometryGetIntersectionProgram](#), [rtProgramCreateFromPTXFile](#), [rtProgramCreateFromPTXString](#)

3.8.3.136 **RTresult** RTAPI [rtGeometrySetPrimitiveCount](#) (**RTgeometry** *geometry*, unsigned int *num_primitives*)

Sets the number of primitives.

Description

[rtGeometrySetPrimitiveCount](#) sets the number of primitives *num_primitives* in *geometry*.

Parameters

| | | |
|----|-----------------------|---|
| in | <i>geometry</i> | The geometry node for which to set the number of primitives |
| in | <i>num_primitives</i> | The number of primitives |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometrySetPrimitiveCount](#) was introduced in OptiX 1.0.

See also [rtGeometryGetPrimitiveCount](#)

3.8.3.137 **RTresult** RTAPI [rtGeometrySetPrimitiveIndexOffset](#) (**RTgeometry** *geometry*, unsigned int *index_offset*)

Sets the primitive index offset.

Description

[rtGeometrySetPrimitiveIndexOffset](#) sets the primitive index offset *index_offset* in *geometry*. In the past, a Geometry object's primitive index range always started at zero (e.g., a Geometry with *N* primitives would have a primitive index range of [0,N-1]). The index offset is used to allow Geometry objects to have primitive index ranges starting at non-zero positions (e.g., a Geometry with *N* primitives and an index offset of *M* would have a primitive index range of [M,M+N-1]). This feature enables the sharing of vertex index buffers between multiple Geometry objects.

Parameters

| | | |
|----|---------------------|---|
| in | <i>geometry</i> | The geometry node for which to set the primitive index offset |
| in | <i>index_offset</i> | The primitive index offset |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGeometrySetPrimitiveIndexOffset](#) was introduced in OptiX 3.5.

See also [rtGeometryGetPrimitiveIndexOffset](#)

3.8.3.138 RTresult RTAPI rtGeometryValidate (RTgeometry *geometry*)

Validates the geometry nodes integrity.

Description

[rtGeometryValidate](#) checks *geometry* for completeness. If *geometry* or any of the objects attached to *geometry* are not valid, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|----|-----------------|-----------------------------------|
| in | <i>geometry</i> | The geometry node to be validated |
|----|-----------------|-----------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtGeometryValidate](#) was introduced in OptiX 1.0.

See also [rtContextValidate](#)

3.8.3.139 RTresult RTAPI rtGetVersion (unsigned int * *version*)

Returns the current OptiX version.

Description

[rtGetVersion](#) returns in *version* a numerically comparable version number of the current OptiX library.

The encoding for the version number prior to OptiX 4.0.0 is major*1000 + minor*10 + micro. For versions 4.0.0 and higher, the encoding is major*10000 + minor*100 + micro. For example, for version 3.5.1 this function would return 3051, and for version 4.5.1 it would return 40501.

Parameters

| | | |
|-----|----------------|----------------------|
| out | <i>version</i> | OptiX version number |
|-----|----------------|----------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGetVersion](#) was introduced in OptiX 1.0.

See also [rtDeviceGetDeviceCount](#)

3.8.3.140 RTresult RTAPI rtGroupCreate (RTcontext *context*, RTgroup * *group*)

Creates a new group.

Description

[rtGroupCreate](#) creates a new group within a context. *context* specifies the target context, and should be a value returned by [rtContextCreate](#). Sets **group* to the handle of a newly created group within *context*. Returns [RT_ERROR_INVALID_VALUE](#) if *group* is *NULL*.

Parameters

| | | |
|-----|----------------|--|
| in | <i>context</i> | Specifies a context within which to create a new group |
| out | <i>group</i> | Returns a newly created group |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGroupCreate](#) was introduced in OptiX 1.0.

See also [rtGroupDestroy](#), [rtContextCreate](#)

3.8.3.141 RTresult RTAPI rtGroupDestroy (RTgroup *group*)

Destroys a group node.

Description

[rtGroupDestroy](#) removes *group* from its context and deletes it. *group* should be a value returned by [rtGroupCreate](#). No child graph nodes are destroyed. After the call, *group* is no longer a valid handle.

Parameters

| | | |
|----|--------------|-------------------------------------|
| in | <i>group</i> | Handle of the group node to destroy |
|----|--------------|-------------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGroupDestroy](#) was introduced in OptiX 1.0.

See also [rtGroupCreate](#)

3.8.3.142 RTresult RTAPI rtGroupGetAcceleration (RTgroup *group*, RTacceleration * *acceleration*)

Returns the acceleration structure attached to a group.

Description

[rtGroupGetAcceleration](#) returns the acceleration structure attached to a group using [rtGroupSetAcceleration](#). If no acceleration structure has previously been set, **acceleration* is set to *NULL*.

Parameters

| | | |
|-----|---------------------|--|
| in | <i>group</i> | The group handle |
| out | <i>acceleration</i> | The returned acceleration structure object |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGroupGetAcceleration](#) was introduced in OptiX 1.0.

See also [rtGroupSetAcceleration](#), [rtAccelerationCreate](#)

3.8.3.143 `RTresult RTAPI rtGroupGetChild (RTgroup group, unsigned int index, RObject * child)`

Returns a child node of a group.

Description

[rtGroupGetChild](#) returns the child object at slot *index* of the parent *group*. If no child has been assigned to the given slot, **child* is set to `NULL`. Returns `RT_ERROR_INVALID_VALUE` if given an invalid child index or `NULL` pointer.

Parameters

| | | |
|-----|--------------|--------------------------------------|
| in | <i>group</i> | The parent group handle |
| in | <i>index</i> | The index of the child slot to query |
| out | <i>child</i> | The returned child object |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGroupGetChild](#) was introduced in OptiX 1.0.

See also [rtGroupSetChild](#), [rtGroupSetChildCount](#), [rtGroupGetChildCount](#), [rtGroupGetChildType](#)

3.8.3.144 `RTresult RTAPI rtGroupGetChildCount (RTgroup group, unsigned int * count)`

Returns the number of child slots for a group.

Description

[rtGroupGetChildCount](#) returns the number of child slots allocated using [rtGroupSetChildCount](#). This includes empty slots which may not yet have actual children assigned by [rtGroupSetChild](#). Returns `RT_ERROR_INVALID_VALUE` if given a `NULL` pointer.

Parameters

| | | |
|-----|--------------|--------------------------------|
| in | <i>group</i> | The parent group handle |
| out | <i>count</i> | Returned number of child slots |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGroupGetChildCount](#) was introduced in OptiX 1.0.

See also [rtGroupSetChild](#), [rtGroupGetChild](#), [rtGroupSetChildCount](#), [rtGroupGetChildType](#)

3.8.3.145 `RTresult RTAPI rtGroupGetChildType (RTgroup group, unsigned int index, RObjecttype * type)`

Get the type of a group child.

Description

[rtGroupGetChildType](#) returns the type of the group child at slot *index*. If no child is associated with the given index, **type* is set to [RT_OBJECTTYPE_UNKNOWN](#) and [RT_ERROR_INVALID_VALUE](#) is returned. Returns [RT_ERROR_INVALID_VALUE](#) if given a *NULL* pointer.

Parameters

| | | |
|-----|--------------|--------------------------------------|
| in | <i>group</i> | The parent group handle |
| in | <i>index</i> | The index of the child slot to query |
| out | <i>type</i> | The returned child type |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGroupGetChildType](#) was introduced in OptiX 1.0.

See also [rtGroupSetChild](#), [rtGroupGetChild](#), [rtGroupSetChildCount](#), [rtGroupGetChildCount](#)

3.8.3.146 RTresult RTAPI rtGroupGetContext (RTgroup *group*, RTcontext * *context*)

Returns the context associated with a group.

Description

[rtGroupGetContext](#) queries a group for its associated context. *group* specifies the group to query, and must be a value returned by [rtGroupCreate](#). Sets **context* to the context associated with *group*.

Parameters

| | | |
|-----|----------------|---|
| in | <i>group</i> | Specifies the group to query |
| out | <i>context</i> | Returns the context associated with the group |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGroupGetContext](#) was introduced in OptiX 1.0.

See also [rtContextCreate](#), [rtGroupCreate](#)

3.8.3.147 RTresult RTAPI rtGroupSetAcceleration (RTgroup *group*, RTacceleration *acceleration*)

Set the acceleration structure for a group.

Description

[rtGroupSetAcceleration](#) attaches an acceleration structure to a group. The acceleration structure must have been previously created using [rtAccelerationCreate](#). Every group is required to have an acceleration structure assigned in order to pass validation. The acceleration structure will be built over the children of the group. For example, if an acceleration structure is attached to a group that has a selector, a geometry group, and a transform child, the acceleration structure will be built over the bounding volumes of these three objects.

Note that it is legal to attach a single [RTacceleration](#) object to multiple groups, as long as the underlying bounds of the children are the same. For example, if another group has three children which are known to have the same bounding volumes as the ones in the example above, the two groups can share an acceleration structure, thus saving build time. This is true even if the details of the children, such as the actual type of a node or its geometry

content, differ from the first set of group children. All that is required is for a child node at a given index to have the same bounds as the other group's child node at the same index.

Sharing an acceleration structure this way corresponds to attaching an acceleration structure to multiple geometry groups at lower graph levels using [rtGeometryGroupSetAcceleration](#).

Parameters

| | | |
|----|---------------------|---|
| in | <i>group</i> | The group handle |
| in | <i>acceleration</i> | The acceleration structure to attach to the group |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGroupSetAcceleration](#) was introduced in OptiX 1.0.

See also [rtGroupGetAcceleration](#), [rtAccelerationCreate](#), [rtGeometryGroupSetAcceleration](#)

3.8.3.148 RTresult RTAPI rtGroupSetChild (RTgroup *group*, unsigned int *index*, RObject *child*)

Attaches a child node to a group.

Description

Attaches a new child node *child* to the parent node *group*. *index* specifies the number of the slot where the child node gets attached. A sufficient number of slots must be allocated using [rtGroupSetChildCount](#). Legal child node types are [RTgroup](#), [RTselector](#), [RTgeometrygroup](#), and [RTtransform](#).

Parameters

| | | |
|----|--------------|--|
| in | <i>group</i> | The parent group handle |
| in | <i>index</i> | The index in the parent's child slot array |
| in | <i>child</i> | The child node to be attached. Can be of type { RTgroup , RTselector , RTgeometrygroup , RTtransform } |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGroupSetChild](#) was introduced in OptiX 1.0.

See also [rtGroupSetChildCount](#), [rtGroupGetChildCount](#), [rtGroupGetChild](#), [rtGroupGetChildType](#)

3.8.3.149 RTresult RTAPI rtGroupSetChildCount (RTgroup *group*, unsigned int *count*)

Sets the number of child nodes to be attached to the group.

Description

[rtGroupSetChildCount](#) specifies the number of child slots in this group. Potentially existing links to children at indices greater than *count-1* are removed. If the call increases the number of slots, the newly created slots are empty and need to be filled using [rtGroupSetChild](#) before validation.

Parameters

| | | |
|----|--------------|---|
| in | <i>group</i> | The parent group handle |
| in | <i>count</i> | Number of child slots to allocate for the group |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGroupSetChildCount](#) was introduced in OptiX 1.0.

See also [rtGroupGetChild](#), [rtGroupGetChildCount](#), [rtGroupGetChildType](#), [rtGroupSetChild](#)

3.8.3.150 RTresult RTAPI rtGroupValidate (RTgroup *group*)

Verifies the state of the group.

Description

[rtGroupValidate](#) checks *group* for completeness. If *group* or any of the objects attached to *group* are not valid, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|----|--------------|-------------------------------------|
| in | <i>group</i> | Specifies the group to be validated |
|----|--------------|-------------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtGroupValidate](#) was introduced in OptiX 1.0.

See also [rtGroupCreate](#)

3.8.3.151 RTresult RTAPI rtMaterialCreate (RTcontext *context*, RTmaterial * *material*)

Creates a new material.

Description

[rtMaterialCreate](#) creates a new material within a context. *context* specifies the target context, as returned by [rtContextCreate](#). Sets **material* to the handle of a newly created material within *context*. Returns [RT_ERROR_INVALID_VALUE](#) if *material* is *NULL*.

Parameters

| | | |
|-----|-----------------|---|
| in | <i>context</i> | Specifies a context within which to create a new material |
| out | <i>material</i> | Returns a newly created material |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtMaterialCreate](#) was introduced in OptiX 1.0.

See also [rtMaterialDestroy](#), [rtContextCreate](#)

3.8.3.152 **RTresult** RTAPI [rtMaterialDeclareVariable](#) (**RTmaterial** *material*, **const char ****name*, **RTvariable ****v*)

Declares a new named variable to be associated with a material.

Description

[rtMaterialDeclareVariable](#) declares a new variable to be associated with a material. *material* specifies the target material, and should be a value returned by [rtMaterialCreate](#). *name* specifies the name of the variable, and should be a *NULL-terminated* string. If there is currently no variable associated with *material* named *name*, and *v* is not *NULL*, a new variable named *name* will be created and associated with *material* and **v* will be set to the handle of the newly-created variable. Otherwise, this call has no effect and returns either [RT_ERROR_INVALID_VALUE](#) if either *name* or *v* is *NULL* or [RT_ERROR_VARIABLE_REDECLARED](#) if *name* is the name of an existing variable associated with the material.

Parameters

| | | |
|-----|-----------------|---|
| in | <i>material</i> | Specifies the material to modify |
| in | <i>name</i> | Specifies the name of the variable |
| out | <i>v</i> | Returns a handle to a newly declared variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_REDECLARED](#)
- [RT_ERROR_ILLEGAL_SYMBOL](#)

History

[rtMaterialDeclareVariable](#) was introduced in OptiX 1.0.

See also [rtMaterialGetVariable](#), [rtMaterialQueryVariable](#), [rtMaterialCreate](#)

3.8.3.153 **RTresult** RTAPI [rtMaterialDestroy](#) (**RTmaterial** *material*)

Destroys a material object.

Description

[rtMaterialDestroy](#) removes *material* from its context and deletes it. *material* should be a value returned by [rtMaterialCreate](#). Associated variables declared via [rtMaterialDeclareVariable](#) are destroyed, but no child graph nodes are destroyed. After the call, *material* is no longer a valid handle.

Parameters

| | | |
|----|-----------------|--|
| in | <i>material</i> | Handle of the material node to destroy |
|----|-----------------|--|

Return values

Relevant return values:

- [RT_SUCCESS](#)

- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtMaterialDestroy](#) was introduced in OptiX 1.0.

See also [rtMaterialCreate](#)

3.8.3.154 **RTresult** RTAPI [rtMaterialGetAnyHitProgram](#) (**RTmaterial** *material*, unsigned int *ray_type_index*, **RTprogram** * *program*)

Returns the any hit program associated with a (material, ray type) tuple.

Description

[rtMaterialGetAnyHitProgram](#) queries the any hit program associated with a (material, ray type) tuple. *material* specifies the material of interest and should be a value returned by [rtMaterialCreate](#). *ray_type_index* specifies the target ray type and should be a value less than the value returned by [rtContextGetRayTypeCount](#). If all parameters are valid, **program* sets to the handle of the any hit program associated with the tuple (*material*, *ray_type_index*). Otherwise, the call has no effect and returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|-----------------------|--|
| in | <i>material</i> | Specifies the material of the (material, ray type) tuple to query |
| in | <i>ray_type_index</i> | Specifies the type of ray of the (material, ray type) tuple to query |
| out | <i>program</i> | Returns the any hit program associated with the (material, ray type) tuple |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtMaterialGetAnyHitProgram](#) was introduced in OptiX 1.0.

See also [rtMaterialSetAnyHitProgram](#), [rtMaterialCreate](#), [rtContextGetRayTypeCount](#)

3.8.3.155 **RTresult** RTAPI [rtMaterialGetClosestHitProgram](#) (**RTmaterial** *material*, unsigned int *ray_type_index*, **RTprogram** * *program*)

Returns the closest hit program associated with a (material, ray type) tuple.

Description

[rtMaterialGetClosestHitProgram](#) queries the closest hit program associated with a (material, ray type) tuple. *material* specifies the material of interest and should be a value returned by [rtMaterialCreate](#). *ray_type_index* specifies the target ray type and should be a value less than the value returned by [rtContextGetRayTypeCount](#). If all parameters are valid, **program* sets to the handle of the any hit program associated with the tuple (*material*, *ray_type_index*). Otherwise, the call has no effect and returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|-----------------------|--|
| in | <i>material</i> | Specifies the material of the (material, ray type) tuple to query |
| in | <i>ray_type_index</i> | Specifies the type of ray of the (material, ray type) tuple to query |
| out | <i>program</i> | Returns the closest hit program associated with the (material, ray type) tuple |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtMaterialGetClosestHitProgram](#) was introduced in OptiX 1.0.

See also [rtMaterialSetClosestHitProgram](#), [rtMaterialCreate](#), [rtContextGetRayTypeCount](#)

3.8.3.156 **RTresult** RTAPI [rtMaterialGetContext](#) (**RTmaterial** *material*, **RTcontext** * *context*)

Returns the context associated with a material.

Description

[rtMaterialGetContext](#) queries a material for its associated context. *material* specifies the material to query, and should be a value returned by [rtMaterialCreate](#). If both parameters are valid, **context* sets to the context associated with *material*. Otherwise, the call has no effect and returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|-----------------|--|
| in | <i>material</i> | Specifies the material to query |
| out | <i>context</i> | Returns the context associated with the material |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtMaterialGetContext](#) was introduced in OptiX 1.0.

See also [rtMaterialCreate](#)

3.8.3.157 **RTresult** RTAPI [rtMaterialGetVariable](#) (**RTmaterial** *material*, unsigned int *index*, **RTvariable** * *v*)

Returns a handle to an indexed variable of a material.

Description

[rtMaterialGetVariable](#) queries the handle of a material's indexed variable. *material* specifies the target material and should be a value returned by [rtMaterialCreate](#). *index* specifies the index of the variable, and should be a value less than [rtMaterialGetVariableCount](#). If *material* is a valid material and *index* is the index of a variable attached to *material*, **v* is set to a handle to that variable. Otherwise, **v* is set to *NULL* and either [RT_ERROR_INVALID_VALUE](#) or [RT_ERROR_VARIABLE_NOT_FOUND](#) is returned depending on the validity of *material*, or *index*, respectively.

Parameters

| | | |
|-----|-----------------|--|
| in | <i>material</i> | Specifies the material to query |
| in | <i>index</i> | Specifies the index of the variable to query |
| out | <i>v</i> | Returns the indexed variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_VARIABLE_NOT_FOUND](#)

History

[rtMaterialGetVariable](#) was introduced in OptiX 1.0.

See also [rtMaterialQueryVariable](#), [rtMaterialGetVariableCount](#), [rtMaterialCreate](#)

3.8.3.158 RTresult RTAPI rtMaterialGetVariableCount (RTmaterial *material*, unsigned int * *count*)

Returns the number of variables attached to a material.

Description

[rtMaterialGetVariableCount](#) queries the number of variables attached to a material. *material* specifies the material, and should be a value returned by [rtMaterialCreate](#). After the call, if both parameters are valid, the number of variables attached to *material* is returned to **count*. Otherwise, the call has no effect and returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|-----------------|---------------------------------|
| in | <i>material</i> | Specifies the material to query |
| out | <i>count</i> | Returns the number of variables |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtMaterialGetVariableCount](#) was introduced in OptiX 1.0.

See also [rtMaterialCreate](#)

3.8.3.159 RTresult RTAPI rtMaterialQueryVariable (RTmaterial *material*, const char * *name*, RTvariable * *v*)

Queries for the existence of a named variable of a material.

Description

[rtMaterialQueryVariable](#) queries for the existence of a material's named variable. *material* specifies the target material and should be a value returned by [rtMaterialCreate](#). *name* specifies the name of the variable, and should be a *NULL-terminated* string. If *material* is a valid material and *name* is the name of a variable attached to *material*, **v* is set to a handle to that variable after the call. Otherwise, **v* is set to *NULL*. If *material* is not a valid material, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|-----------------|---|
| in | <i>material</i> | Specifies the material to query |
| in | <i>name</i> | Specifies the name of the variable to query |
| out | <i>v</i> | Returns a the named variable, if it exists |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtMaterialQueryVariable](#) was introduced in OptiX 1.0.

See also [rtMaterialGetVariable](#), [rtMaterialCreate](#)

3.8.3.160 RTresult RTAPI rtMaterialRemoveVariable (RTmaterial *material*, RTvariable *v*)

Removes a variable from a material.

Description

[rtMaterialRemoveVariable](#) removes a variable from a material. The material of interest is specified by *material*, which should be a value returned by [rtMaterialCreate](#). The variable to remove is specified by *v*, which should be a value returned by [rtMaterialDeclareVariable](#). Once a variable has been removed from this material, another variable with the same name as the removed variable may be declared. If *material* does not refer to a valid material, this call has no effect and returns [RT_ERROR_INVALID_VALUE](#). If *v* is not a valid variable or does not belong to *material*, this call has no effect and returns [RT_ERROR_INVALID_VALUE](#) or [RT_ERROR_VARIABLE_NOT_FOUND](#), respectively.

Parameters

| | | |
|----|-----------------|----------------------------------|
| in | <i>material</i> | Specifies the material to modify |
| in | <i>v</i> | Specifies the variable to remove |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_NOT_FOUND](#)

History

[rtMaterialRemoveVariable](#) was introduced in OptiX 1.0.

See also [rtMaterialDeclareVariable](#), [rtMaterialCreate](#)

3.8.3.161 RTresult RTAPI rtMaterialSetAnyHitProgram (RTmaterial *material*, unsigned int *ray_type_index*, RTprogram *program*)

Sets the any hit program associated with a (material, ray type) tuple.

Description

[rtMaterialSetAnyHitProgram](#) specifies an any hit program to associate with a (material, ray type) tuple. *material* specifies the target material and should be a value returned by [rtMaterialCreate](#). *ray_type_index* specifies the type of ray to which the program applies and should be a value less than the value returned by [rtContextGetRayTypeCount](#). *program* specifies the target any hit program which applies to the tuple (*material*, *ray_type_index*) and should be a value returned by either [rtProgramCreateFromPTXString](#) or [rtProgramCreateFromPTXFile](#).

Parameters

| | | |
|----|-----------------------|--|
| in | <i>material</i> | Specifies the material of the (material, ray type) tuple to modify |
| in | <i>ray_type_index</i> | Specifies the type of ray of the (material, ray type) tuple to modify |
| in | <i>program</i> | Specifies the any hit program to associate with the (material, ray type) tuple |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

[rtMaterialSetAnyHitProgram](#) was introduced in OptiX 1.0.

See also [rtMaterialGetAnyHitProgram](#), [rtMaterialCreate](#), [rtContextGetRayTypeCount](#), [rtProgramCreateFromPTXString](#), [rtProgramCreateFromPTXFile](#)

3.8.3.162 **RTresult** RTAPI [rtMaterialSetClosestHitProgram](#) (**RTmaterial** *material*, unsigned int *ray_type_index*, **RTprogram** *program*)

Sets the closest hit program associated with a (material, ray type) tuple.

Description

[rtMaterialSetClosestHitProgram](#) specifies a closest hit program to associate with a (material, ray type) tuple. *material* specifies the material of interest and should be a value returned by [rtMaterialCreate](#). *ray_type_index* specifies the type of ray to which the program applies and should be a value less than the value returned by [rtContextGetRayTypeCount](#). *program* specifies the target closest hit program which applies to the tuple (*material*, *ray_type_index*) and should be a value returned by either [rtProgramCreateFromPTXString](#) or [rtProgramCreateFromPTXFile](#).

Parameters

| | | |
|----|-----------------------|--|
| in | <i>material</i> | Specifies the material of the (material, ray type) tuple to modify |
| in | <i>ray_type_index</i> | Specifies the ray type of the (material, ray type) tuple to modify |
| in | <i>program</i> | Specifies the closest hit program to associate with the (material, ray type) tuple |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

[rtMaterialSetClosestHitProgram](#) was introduced in OptiX 1.0.

See also [rtMaterialGetClosestHitProgram](#), [rtMaterialCreate](#), [rtContextGetRayTypeCount](#), [rtProgramCreateFromPTXString](#), [rtProgramCreateFromPTXFile](#)

3.8.3.163 **RTresult** RTAPI [rtMaterialValidate](#) (**RTmaterial** *material*)

Verifies the state of a material.

Description

[rtMaterialValidate](#) checks *material* for completeness. If *material* or any of the objects attached to *material* are not valid, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|----|-----------------|--|
| in | <i>material</i> | Specifies the material to be validated |
|----|-----------------|--|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtMaterialValidate](#) was introduced in OptiX 1.0.

See also [rtMaterialCreate](#)

3.8.3.164 `RTresult RTAPI rtProgramCreateFromPTXFile (RTcontext context, const char * filename, const char * program_name, RTprogram * program)`

Creates a new program object.

Description

[rtProgramCreateFromPTXFile](#) allocates and returns a handle to a new program object. The program is created from PTX code held in *filename* from function *program_name*.

Parameters

| | | |
|----|---------------------|---|
| in | <i>context</i> | The context to create the program in |
| in | <i>filename</i> | Path to the file containing the PTX code |
| in | <i>program_name</i> | The name of the PTX function to create the program from |
| in | <i>program</i> | Handle to the program to be created |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_INVALID_SOURCE](#)
- [RT_ERROR_FILE_NOT_FOUND](#)

History

[rtProgramCreateFromPTXFile](#) was introduced in OptiX 1.0.

See also [RT_PROGRAM](#), [rtProgramCreateFromPTXString](#), [rtProgramDestroy](#)

3.8.3.165 `RTresult RTAPI rtProgramCreateFromPTXString (RTcontext context, const char * ptx, const char * program_name, RTprogram * program)`

Creates a new program object.

Description

[rtProgramCreateFromPTXString](#) allocates and returns a handle to a new program object. The program is created from PTX code held in the *NULL-terminated* string *ptx* from function *program_name*.

Parameters

| | | |
|----|---------------------|---|
| in | <i>context</i> | The context to create the program in |
| in | <i>ptx</i> | The string containing the PTX code |
| in | <i>program_name</i> | The name of the PTX function to create the program from |
| in | <i>program</i> | Handle to the program to be created |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)

- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_INVALID_SOURCE](#)

History

[rtProgramCreateFromPTXString](#) was introduced in OptiX 1.0.

See also [RT_PROGRAM](#), [rtProgramCreateFromPTXFile](#), [rtProgramDestroy](#)

3.8.3.166 `RTresult RTAPI rtProgramDeclareVariable (RTprogram program, const char * name, RTvariable * v)`

Declares a new named variable associated with a program.

Description

[rtProgramDeclareVariable](#) declares a new variable, *name*, and associates it with the program. A variable can only be declared with the same name once on the program. Any attempt to declare multiple variables with the same name will cause the call to fail and return [RT_ERROR_VARIABLE_REDECLARED](#). If *name* or *v* is *NULL* returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|----------------|---|
| in | <i>program</i> | The program the declared variable will be attached to |
| in | <i>name</i> | The name of the variable to be created |
| out | <i>v</i> | Return handle to the variable to be created |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_REDECLARED](#)
- [RT_ERROR_ILLEGAL_SYMBOL](#)

History

[rtProgramDeclareVariable](#) was introduced in OptiX 1.0.

See also [rtProgramRemoveVariable](#), [rtProgramGetVariable](#), [rtProgramGetVariableCount](#), [rtProgramQueryVariable](#)

3.8.3.167 `RTresult RTAPI rtProgramDestroy (RTprogram program)`

Destroys a program object.

Description

[rtProgramDestroy](#) removes *program* from its context and deletes it. *program* should be a value returned by *rtProgramCreate**. Associated variables declared via [rtProgramDeclareVariable](#) are destroyed. After the call, *program* is no longer a valid handle.

Parameters

| | | |
|----|----------------|----------------------------------|
| in | <i>program</i> | Handle of the program to destroy |
|----|----------------|----------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtProgramDestroy](#) was introduced in OptiX 1.0.

See also [rtProgramCreateFromPTXFile](#), [rtProgramCreateFromPTXString](#)

3.8.3.168 **RTresult** RTAPI [rtProgramGetContext](#) (**RTprogram** *program*, **RTcontext** * *context*)

Gets the context object that created a program.

Description

[rtProgramGetContext](#) returns a handle to the context object that was used to create *program*. Returns [RT_ERROR_INVALID_VALUE](#) if *context* is *NULL*.

Parameters

| | | |
|-----|----------------|--|
| in | <i>program</i> | The program to be queried for its context object |
| out | <i>context</i> | The return handle for the requested context object |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtProgramGetContext](#) was introduced in OptiX 1.0.

See also [rtContextCreate](#)

3.8.3.169 **RTresult** RTAPI [rtProgramGetId](#) (**RTprogram** *program*, int * *program_id*)

Returns the ID for the Program object.

Description

[rtProgramGetId](#) returns an ID for the provided program. The returned ID is used to reference *program* from device code. If *program_id* is *NULL* or the *program* is not a valid *RTprogram*, returns [RT_ERROR_INVALID_VALUE](#). [RT_PROGRAM_ID_NULL](#) can be used as a sentinel for a non-existent program, since this value will never be returned as a valid program id.

Parameters

| | | |
|-----|-------------------|--------------------------------------|
| in | <i>program</i> | The program to be queried for its id |
| out | <i>program_id</i> | The returned ID of the program. |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtProgramGetId](#) was introduced in OptiX 3.6.

See also [rtContextGetProgramFromId](#)

3.8.3.170 **RTresult** RTAPI [rtProgramGetVariable](#) (**RTprogram** *program*, unsigned int *index*, **RTvariable** * *v*)

Returns a handle to a variable attached to a program by index.

Description

[rtProgramGetVariable](#) returns a handle to a variable in **v* attached to *program* with [rtProgramDeclareVariable](#) by *index*. *index* must be between 0 and one less than the value returned by [rtProgramGetVariableCount](#). The order in which variables are enumerated is not constant and may change as variables are attached and removed from the program object.

Parameters

| | | |
|-----|----------------|---|
| in | <i>program</i> | The program to be queried for the indexed variable object |
| in | <i>index</i> | The index of the variable to return |
| out | <i>v</i> | Return handle to the variable object specified by the index |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_NOT_FOUND](#)

History

[rtProgramGetVariable](#) was introduced in OptiX 1.0.

See also [rtProgramDeclareVariable](#), [rtProgramRemoveVariable](#), [rtProgramGetVariableCount](#), [rtProgramQueryVariable](#)

3.8.3.171 **RTresult** RTAPI [rtProgramGetVariableCount](#) (**RTprogram** *program*, unsigned int * *count*)

Returns the number of variables attached to a program.

Description

[rtProgramGetVariableCount](#) returns, in **count*, the number of variable objects that have been attached to *program*.

Parameters

| | | |
|-----|----------------|--|
| in | <i>program</i> | The program to be queried for its variable count |
| out | <i>count</i> | The return handle for the number of variables attached to this program |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtProgramGetVariableCount](#) was introduced in OptiX 1.0.

See also [rtProgramDeclareVariable](#), [rtProgramRemoveVariable](#), [rtProgramGetVariable](#), [rtProgramQueryVariable](#)

3.8.3.172 **RTresult** RTAPI [rtProgramQueryVariable](#) (**RTprogram** *program*, **const char *** *name*, **RTvariable *** *v*)

Returns a handle to the named variable attached to a program.

Description

[rtProgramQueryVariable](#) returns a handle to a variable object, in **v*, attached to *program* referenced by the *NULL-terminated* string *name*. If *name* is not the name of a variable attached to *program*, **v* will be *NULL* after the call.

Parameters

| | | |
|-----|----------------|--|
| in | <i>program</i> | The program to be queried for the named variable |
| in | <i>name</i> | The name of the program to be queried for |
| out | <i>v</i> | The return handle to the variable object |
| | <i>program</i> | Handle to the program to be created |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtProgramQueryVariable](#) was introduced in OptiX 1.0.

See also [rtProgramDeclareVariable](#), [rtProgramRemoveVariable](#), [rtProgramGetVariable](#), [rtProgramGetVariableCount](#)

3.8.3.173 **RTresult** RTAPI [rtProgramRemoveVariable](#) (**RTprogram** *program*, **RTvariable** *v*)

Removes the named variable from a program.

Description

[rtProgramRemoveVariable](#) removes variable *v* from the *program* object. Once a variable has been removed from this program, another variable with the same name as the removed variable may be declared.

Parameters

| | | |
|----|----------------|---|
| in | <i>program</i> | The program to remove the variable from |
| in | <i>v</i> | The variable to remove |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_NOT_FOUND](#)

History

[rtProgramRemoveVariable](#) was introduced in OptiX 1.0.

See also [rtProgramDeclareVariable](#), [rtProgramGetVariable](#), [rtProgramGetVariableCount](#), [rtProgramQueryVariable](#)

3.8.3.174 **RTresult** RTAPI **rtProgramValidate** (**RTprogram** *program*)

Validates the state of a program.

Description

[rtProgramValidate](#) checks *program* for completeness. If *program* or any of the objects attached to *program* are not valid, returns [RT_ERROR_INVALID_CONTEXT](#).

Parameters

| | | |
|----|----------------|-----------------------------|
| in | <i>program</i> | The program to be validated |
|----|----------------|-----------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtProgramValidate](#) was introduced in OptiX 1.0.

See also [rtProgramCreateFromPTXFile](#), [rtProgramCreateFromPTXString](#)

3.8.3.175 **RTresult** RTAPI **rtRemoteDeviceCreate** (**const char ****url*, **const char ****username*, **const char ****password*, **RTremotedevice ****remote_dev*)

Create a device for remote rendering on VCAs.

Description

Establishes a connection to a remote OptiX device, e.g. a VCA or cluster of VCAs. This opens a connection to the cluster manager software running at *address*, using *username* and *password* as authentication strings. *address* is a WebSocket URL of the form "ws://localhost:80" or "wss://localhost:443", *username* and *password* as plain text strings for authenticating on the remote device. If successful, it initializes a new [RTremotedevice](#) object.

In order to use this newly created remote device, a rendering instance needs to be configured by selecting a software configuration and reserving a number of nodes in the VCA. See [rtRemoteDeviceReserve](#) for more details.

After a rendering instance is properly initialized, a remote device must be associated with a context to be used. Calling `rtContextSetDevices` creates this association. Any further OptiX calls will be directed to the remote device.

Parameters

| | | |
|-----|-------------------|--|
| in | <i>url</i> | The WebSocket URL to connect to |
| in | <i>username</i> | Username in plain text |
| in | <i>password</i> | Password in plain text |
| out | <i>remote_dev</i> | A handle to the new remote device object |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_CONNECTION_FAILED](#)
- [RT_ERROR_AUTHENTICATION_FAILED](#)

History

[rtRemoteDeviceCreate](#) was introduced in OptiX 3.8.

See also [rtRemoteDeviceDestroy](#) [rtRemoteDeviceGetAttribute](#) [rtRemoteDeviceReserve](#) [rtRemoteDeviceRelease](#) [rtContextSetRemoteDevice](#)

3.8.3.176 **RTresult** RTAPI [rtRemoteDeviceDestroy](#) (**RTremotedevice** *remote_dev*)

Destroys a remote device.

Description

Closes the network connection to the remote device and destroys the corresponding [RTremotedevice](#) object.

Parameters

| | | |
|----|-------------------|-------------------------------------|
| in | <i>remote_dev</i> | The remote device object to destroy |
|----|-------------------|-------------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtRemoteDeviceDestroy](#) was introduced in OptiX 3.8.

See also [rtRemoteDeviceCreate](#) [rtRemoteDeviceGetAttribute](#) [rtRemoteDeviceReserve](#) [rtRemoteDeviceRelease](#) [rtContextSetRemoteDevice](#)

3.8.3.177 **RTresult** RTAPI [rtRemoteDeviceGetAttribute](#) (**RTremotedevice** *remote_dev*, **RTremotedeviceattribute** *attrib*, **RTsize** *size*, void * *p*)

Queries attributes of a remote device.

Description

In order to gather information about a remote device, several attributes can be queried through [rtRemoteDeviceGetAttribute](#).

Each attribute can have a different size. The sizes are given in the following list:

- [RT_REMOTEDEVICE_ATTRIBUTE_CLUSTER_URL](#) size of provided destination buffer
- [RT_REMOTEDEVICE_ATTRIBUTE_HEAD_NODE_URL](#) size of provided destination buffer

- [RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS](#) sizeof(int)
- [RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS](#) size of provided destination buffer
- [RT_REMOTEDEVICE_ATTRIBUTE_STATUS](#) sizeof(RTremotedevicestatus)
- [RT_REMOTEDEVICE_ATTRIBUTE_NUM_TOTAL_NODES](#) sizeof(int)
- [RT_REMOTEDEVICE_ATTRIBUTE_NUM_FREE_NODES](#) sizeof(int)
- [RT_REMOTEDEVICE_ATTRIBUTE_NUM_RESERVED_NODES](#) sizeof(int)
- [RT_REMOTEDEVICE_ATTRIBUTE_NAME](#) size of provided destination buffer
- [RT_REMOTEDEVICE_ATTRIBUTE_NUM_GPUS](#) sizeof(int)
- [RT_REMOTEDEVICE_ATTRIBUTE_GPU_TOTAL_MEMORY](#) sizeof(RTsize)

The following attributes can be queried when a remote device is connected:

- [RT_REMOTEDEVICE_ATTRIBUTE_CLUSTER_URL](#)
- [RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS](#)
- [RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS](#)
- [RT_REMOTEDEVICE_ATTRIBUTE_STATUS](#)
- [RT_REMOTEDEVICE_ATTRIBUTE_NUM_TOTAL_NODES](#)
- [RT_REMOTEDEVICE_ATTRIBUTE_NUM_FREE_NODES](#)
- [RT_REMOTEDEVICE_ATTRIBUTE_NAME](#)
- [RT_REMOTEDEVICE_ATTRIBUTE_GPU_TOTAL_MEMORY](#)

The following attributes require a valid reservation to be queried:

- [RT_REMOTEDEVICE_ATTRIBUTE_HEAD_NODE_URL](#)
- [RT_REMOTEDEVICE_ATTRIBUTE_NUM_RESERVED_NODES](#)
- [RT_REMOTEDEVICE_ATTRIBUTE_NUM_GPUS](#)

[RT_REMOTEDEVICE_ATTRIBUTE_CLUSTER_URL](#) The URL of the Cluster Manager associated with this remote device.

[RT_REMOTEDEVICE_ATTRIBUTE_HEAD_NODE_URL](#) The URL of the rendering instance being used, once it has been reserved and initialized.

[RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS](#) Number of compatible software configurations available in the remote device.

[RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS](#) Base entry for a list of compatible software configurations in the device. A configuration is a text description for a software package installed in the remote device, intended as a guide to the user in selecting from the pool of compatible configurations. This list is already filtered and it only contains entries on the remote device compatible with the client library being used. Each entry can be accessed as the attribute ([RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS](#) + index), with index being zero-based. The configuration description for the given index is copied into the destination buffer. A suggested size for the destination buffer is 256 characters. The number of entries in the list is given by the value of [RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS](#). Only configurations compatible with the client version being used are listed.

[RT_REMOTEDEVICE_ATTRIBUTE_STATUS](#) Returns the current status of the remote device, as one of the following:

- [RT_REMOTEDEVICE_STATUS_READY](#) The remote device is ready for use.
- [RT_REMOTEDEVICE_STATUS_CONNECTED](#) The remote device is connected to a cluster manager, but no reservation exists.
- [RT_REMOTEDEVICE_STATUS_RESERVED](#) The remote device has a rendering instance reserved, but it is not yet ready.
- [RT_REMOTEDEVICE_STATUS_DISCONNECTED](#) The remote device has disconnected.

[RT_REMOTEDEVICE_ATTRIBUTE_NUM_TOTAL_NODES](#) Total number of nodes in the cluster of VCAs.

[RT_REMOTEDEVICE_ATTRIBUTE_NUM_FREE_NODES](#) Number of free nodes available.

[RT_REMOTEDEVICE_ATTRIBUTE_NUM_RESERVED_NODES](#) Number of nodes used by the current reservation.

[RT_REMOTEDEVICE_ATTRIBUTE_NUM_GPUS](#) Number of GPUs used by the current reservation.

[RT_REMOTEDEVICE_ATTRIBUTE_NAME](#) Common name assigned the Remote Device.

[RT_REMOTEDEVICE_ATTRIBUTE_GPU_TOTAL_MEMORY](#) Total amount of memory on each GPU, in bytes.

Parameters

| | | |
|-----------|-------------------|----------------------------|
| <i>in</i> | <i>remote_dev</i> | The remote device to query |
|-----------|-------------------|----------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtRemoteDeviceGetAttribute](#) was introduced in OptiX 3.8.

See also [rtRemoteDeviceCreate](#) [rtRemoteDeviceReserve](#) [rtRemoteDeviceRelease](#) [rtContextSetRemoteDevice](#)

3.8.3.178 `RTresult RTAPI rtRemoteDeviceRelease (RTremotedevice remote_dev)`

Release reserved nodes on a remote device.

Description

Releases an existing reservation on the remote device. The rendering instance on the remote device is destroyed, and all its remote context information is lost. Further OptiX calls will no longer be directed to the device. A new reservation can take place.

Parameters

| | | |
|-----------|-------------------|---|
| <i>in</i> | <i>remote_dev</i> | The remote device on which the reservation was made |
|-----------|-------------------|---|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtRemoteDeviceRelease](#) was introduced in OptiX 3.8.

See also [rtRemoteDeviceCreate](#) [rtRemoteDeviceGetAttribute](#) [rtRemoteDeviceReserve](#) [rtContextSetRemoteDevice](#)

3.8.3.179 `RTresult RTAPI rtRemoteDeviceReserve (RTremotedevice remote_dev, unsigned int num_nodes, unsigned int configuration)`

Reserve nodes for rendering on a remote device.

Description

Reserves nodes in the remote device to form a rendering instance. Receives *num_nodes* as the number of nodes to reserve, and *configuration* as the index of the software package to use for the created instance. Both the number of available nodes and the list of available configurations in a remote device can be retrieved by [rtRemoteDeviceGetAttribute](#).

After successfully reserving the nodes, the [RT_REMOTEDEVICE_ATTRIBUTE_STATUS](#) attribute should be polled repeatedly. The rendering instance is ready for use when that attribute is set to `RT_REMOTE_DEVICE_STATUS_READY`.

Only a single reservation per remote device and user can exist at any given time (i.e. a user can have only one rendering instance per remote device). This includes reservations performed through other means, like previous runs that were not properly released, or manual reservations over the cluster manager web interface.

Parameters

| | | |
|----|----------------------|--|
| in | <i>remote_dev</i> | The remote device on which to reserve nodes |
| in | <i>num_nodes</i> | The number of nodes to reserve |
| in | <i>configuration</i> | The index of the software configuration to use |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtRemoteDeviceReserve](#) was introduced in OptiX 3.8.

See also [rtRemoteDeviceCreate](#) [rtRemoteDeviceGetAttribute](#) [rtRemoteDeviceRelease](#) [rtContextSetRemoteDevice](#)

3.8.3.180 `RTresult RTAPI rtSelectorCreate (RTcontext context, RTselector * selector)`

Creates a Selector node.

Description

Creates a new Selector node within *context*. After calling [rtSelectorCreate](#) the new node is in a "raw" state. For the node to be functional, a visit program must be assigned using [rtSelectorSetVisitProgram](#). Furthermore, a number of (zero or more) children can be attached by using [rtSelectorSetChildCount](#) and [rtSelectorSetChild](#). Sets **selector* to the handle of a newly created selector within *context*. Returns [RT_ERROR_INVALID_VALUE](#) if *selector* is `NULL`.

Parameters

| | | |
|-----|-----------------|--|
| in | <i>context</i> | Specifies the rendering context of the Selector node |
| out | <i>selector</i> | New Selector node handle |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorCreate](#) was introduced in OptiX 1.0.

See also [rtSelectorDestroy](#), [rtSelectorValidate](#), [rtSelectorGetContext](#), [rtSelectorSetVisitProgram](#), [rtSelectorSetChildCount](#), [rtSelectorSetChild](#)

3.8.3.181 **RTresult** RTAPI [rtSelectorDeclareVariable](#) (**RTselector** *selector*, const char * *name*, **RTvariable** * *v*)

Declares a variable associated with a Selector node.

Description

Declares a new variable identified by *name*, and associates it with the Selector node *selector*. The new variable handle is returned in *v*. After declaration, a variable does not have a type until its value is set by an [rtVariableSet{...}](#) function. Once a variable type has been set, it cannot be changed, i.e., only [rtVariableSet{...}](#) functions of the same type can be used to change the value of the variable.

Parameters

| | | |
|-----|-----------------|----------------------|
| in | <i>selector</i> | Selector node handle |
| in | <i>name</i> | Variable identifier |
| out | <i>v</i> | New variable handle |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_REDECLARED](#)
- [RT_ERROR_ILLEGAL_SYMBOL](#)

History

[rtSelectorDeclareVariable](#) was introduced in OptiX 1.0.

See also [rtSelectorQueryVariable](#), [rtSelectorRemoveVariable](#), [rtSelectorGetVariableCount](#), [rtSelectorGetVariable](#), [rtVariableSet{...}](#)

3.8.3.182 **RTresult** RTAPI [rtSelectorDestroy](#) (**RTselector** *selector*)

Destroys a selector node.

Description

[rtSelectorDestroy](#) removes *selector* from its context and deletes it. *selector* should be a value returned by [rtSelectorCreate](#). Associated variables declared via [rtSelectorDeclareVariable](#) are destroyed, but no child graph nodes are destroyed. After the call, *selector* is no longer a valid handle.

Parameters

| | | |
|----|-----------------|--|
| in | <i>selector</i> | Handle of the selector node to destroy |
|----|-----------------|--|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorDestroy](#) was introduced in OptiX 1.0.

See also [rtSelectorCreate](#), [rtSelectorValidate](#), [rtSelectorGetContext](#)

3.8.3.183 `RTresult RTAPI rtSelectorGetChild (RTselector selector, unsigned int index, RObject * child)`

Returns a child node that is attached to a Selector node.

Description

[rtSelectorGetChild](#) returns in *child* a handle of the child node currently attached to *selector* at slot *index*. The index value must be lower than the number previously set by [rtSelectorSetChildCount](#), thus it must be in the range from 0 to [rtSelectorGetChildCount](#) - 1. The returned pointer is of generic type [RObject](#) and needs to be cast to the actual child type, which can be [RTgroup](#), [RTselector](#), [RTgeometrygroup](#), or [RTtransform](#). The actual type of *child* can be queried using [rtSelectorGetChildType](#);

Parameters

| | | |
|-----|-----------------|--|
| in | <i>selector</i> | Selector node handle |
| in | <i>index</i> | Child node index |
| out | <i>child</i> | Child node handle. Can be { RTgroup , RTselector , RTgeometrygroup , RTtransform } |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorGetChild](#) was introduced in OptiX 1.0.

See also [rtSelectorSetChildCount](#), [rtSelectorGetChildCount](#), [rtSelectorSetChild](#), [rtSelectorGetChildType](#)

3.8.3.184 `RTresult RTAPI rtSelectorGetChildCount (RTselector selector, unsigned int * count)`

Returns the number of child node slots of a Selector node.

Description

[rtSelectorGetChildCount](#) returns in *count* the number of child node slots that have been previously reserved for the Selector node *selector* by [rtSelectorSetChildCount](#). The value of *count* does not reflect the actual number of child nodes that have so far been attached to the Selector node using [rtSelectorSetChild](#).

Parameters

| | | |
|-----|-----------------|---|
| in | <i>selector</i> | Selector node handle |
| out | <i>count</i> | Number of child node slots reserved for <i>selector</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorGetChildCount](#) was introduced in OptiX 1.0.

See also [rtSelectorSetChildCount](#), [rtSelectorSetChild](#), [rtSelectorGetChild](#), [rtSelectorGetChildType](#)

3.8.3.185 `RTresult RTAPI rtSelectorGetChildType (RTselector selector, unsigned int index, RObjecttype * type)`

Returns type information about a Selector child node.

Description

[rtSelectorGetChildType](#) queries the type of the child node attached to *selector* at slot *index*. If no child is associated with the given index, **type* is set to [RT_OBJECTTYPE_UNKNOWN](#) and [RT_ERROR_INVALID_VALUE](#) is returned. Returns [RT_ERROR_INVALID_VALUE](#) if given a *NULL* pointer. The returned type is one of:

[RT_OBJECTTYPE_GROUP](#) [RT_OBJECTTYPE_GEOMETRY_GROUP](#) [RT_OBJECTTYPE_TRANSFORM](#) [RT_OBJECTTYPE_SELECTOR](#)

Parameters

| | | |
|-----|-----------------|------------------------|
| in | <i>selector</i> | Selector node handle |
| in | <i>index</i> | Child node index |
| out | <i>type</i> | Type of the child node |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorGetChildType](#) was introduced in OptiX 1.0.

See also [rtSelectorSetChildCount](#), [rtSelectorGetChildCount](#), [rtSelectorSetChild](#), [rtSelectorGetChild](#)

3.8.3.186 `RTresult RTAPI rtSelectorGetContext (RTselector selector, RTcontext * context)`

Returns the context of a Selector node.

Description

[rtSelectorGetContext](#) returns in *context* the rendering context in which the Selector node *selector* has been created.

Parameters

| | | |
|-----|-----------------|---|
| in | <i>selector</i> | Selector node handle |
| out | <i>context</i> | The context, <i>selector</i> belongs to |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorGetContext](#) was introduced in OptiX 1.0.

See also [rtSelectorCreate](#), [rtSelectorDestroy](#), [rtSelectorValidate](#)

3.8.3.187 **RTresult** RTAPI [rtSelectorGetVariable](#) (**RTselector** *selector*, unsigned int *index*, **RTvariable** * *v*)

Returns a variable associated with a Selector node.

Description

Returns in *v* a handle to the variable located at position *index* in the Selectors's variable array. *index* is a sequential number depending on the order of variable declarations. The index must be in the range from 0 to [rtSelectorGetVariableCount](#) - 1. The current value of a variable can be retrieved from its handle by using an appropriate [rtVariableGet{...}](#) function matching the variable's type.

Parameters

| | | |
|-----|-----------------|----------------------|
| in | <i>selector</i> | Selector node handle |
| in | <i>index</i> | Variable index |
| out | <i>v</i> | Variable handle |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorGetVariable](#) was introduced in OptiX 1.0.

See also [rtSelectorDeclareVariable](#), [rtSelectorQueryVariable](#), [rtSelectorRemoveVariable](#), [rtSelectorGetVariableCount](#), [rtVariableGet{...}](#)

3.8.3.188 **RTresult** RTAPI [rtSelectorGetVariableCount](#) (**RTselector** *selector*, unsigned int * *count*)

Returns the number of variables attached to a Selector node.

Description

[rtSelectorGetVariableCount](#) returns in *count* the number of variables that are currently attached to the Selector node *selector*.

Parameters

| | | |
|-----|-----------------|---|
| in | <i>selector</i> | Selector node handle |
| out | <i>count</i> | Number of variables associated with <i>selector</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorGetVariableCount](#) was introduced in OptiX 1.0.

See also [rtSelectorDeclareVariable](#), [rtSelectorQueryVariable](#), [rtSelectorRemoveVariable](#), [rtSelectorGetVariable](#)

3.8.3.189 **RTresult** RTAPI [rtSelectorGetVisitProgram](#) (**RTselector** *selector*, **RTprogram** * *program*)

Returns the currently assigned visit program.

Description

[rtSelectorGetVisitProgram](#) returns in *program* a handle of the visit program currently bound to *selector*.

Parameters

| | | |
|-----|-----------------|---|
| in | <i>selector</i> | Selector node handle |
| out | <i>program</i> | Current visit program assigned to <i>selector</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorGetVisitProgram](#) was introduced in OptiX 1.0.

See also [rtSelectorSetVisitProgram](#)

3.8.3.190 **RTresult** RTAPI [rtSelectorQueryVariable](#) (**RTselector** *selector*, **const char** * *name*, **RTvariable** * *v*)

Returns a variable associated with a Selector node.

Description

Returns in *v* a handle to the variable identified by *name*, which is associated with the Selector node *selector*. The current value of a variable can be retrieved from its handle by using an appropriate *rtVariableGet{...}* function matching the variable's type.

Parameters

| | | |
|-----|-----------------|----------------------|
| in | <i>selector</i> | Selector node handle |
| in | <i>name</i> | Variable identifier |
| out | <i>v</i> | Variable handle |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorQueryVariable](#) was introduced in OptiX 1.0.

See also [rtSelectorDeclareVariable](#), [rtSelectorRemoveVariable](#), [rtSelectorGetVariableCount](#), [rtSelectorGetVariable](#), [rtVariableGet{...}](#)

3.8.3.191 **RTresult** RTAPI rtSelectorRemoveVariable (**RTselector** *selector*, **RTvariable** *v*)

Removes a variable from a Selector node.

Description

[rtSelectorRemoveVariable](#) removes the variable *v* from the Selector node *selector* and deletes it. The handle *v* must be considered invalid afterwards.

Parameters

| | | |
|----|-----------------|----------------------|
| in | <i>selector</i> | Selector node handle |
| in | <i>v</i> | Variable handle |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_VARIABLE_NOT_FOUND](#)

History

[rtSelectorRemoveVariable](#) was introduced in OptiX 1.0.

See also [rtSelectorDeclareVariable](#), [rtSelectorQueryVariable](#), [rtSelectorGetVariableCount](#), [rtSelectorGetVariable](#)

3.8.3.192 **RTresult** RTAPI rtSelectorSetChild (**RTselector** *selector*, unsigned int *index*, **RtObject** *child*)

Attaches a child node to a Selector node.

Description

Attaches a new child node *child* to the parent node *selector*. *index* specifies the number of the slot where the child node gets attached. The index value must be lower than the number previously set by [rtSelectorSetChildCount](#), thus it must be in the range from 0 to [rtSelectorGetChildCount](#) - 1. Legal child node types are [RTgroup](#), [RTselector](#), [RTgeometrygroup](#), and [RTtransform](#).

Parameters

| | | |
|----|-----------------|--|
| in | <i>selector</i> | Selector node handle |
| in | <i>index</i> | Index of the parent slot the node <i>child</i> gets attached to |
| in | <i>child</i> | Child node to be attached. Can be { RTgroup , RTselector , RTgeometrygroup , RTtransform } |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorSetChild](#) was introduced in OptiX 1.0.

See also [rtSelectorSetChildCount](#), [rtSelectorGetChildCount](#), [rtSelectorGetChild](#), [rtSelectorGetChildType](#)

3.8.3.193 `RTresult RTAPI rtSelectorSetChildCount (RTselector selector, unsigned int count)`

Specifies the number of child nodes to be attached to a Selector node.

Description

[rtSelectorSetChildCount](#) allocates a number of children slots, i.e., it pre-defines the exact number of child nodes the parent Selector node *selector* will have. Child nodes have to be attached to the Selector node using [rtSelectorSetChild](#). Empty slots will cause a validation error.

Parameters

| | | |
|----|-----------------|---|
| in | <i>selector</i> | Selector node handle |
| in | <i>count</i> | Number of child nodes to be attached to <i>selector</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorSetChildCount](#) was introduced in OptiX 1.0.

See also [rtSelectorValidate](#), [rtSelectorGetChildCount](#), [rtSelectorSetChild](#), [rtSelectorGetChild](#), [rtSelectorGetChildType](#)

3.8.3.194 `RTresult RTAPI rtSelectorSetVisitProgram (RTselector selector, RTprogram program)`

Assigns a visit program to a Selector node.

Description

[rtSelectorSetVisitProgram](#) specifies a visit program that is executed when the Selector node *selector* gets visited by a ray during traversal of the model graph. A visit program steers how traversal of the Selector's children is performed. It usually chooses only a single child to continue traversal, but is also allowed to process zero or multiple children. Programs can be created from PTX files using [rtProgramCreateFromPTXFile](#).

Parameters

| | | |
|----|-----------------|--|
| in | <i>selector</i> | Selector node handle |
| in | <i>program</i> | Program handle associated with a visit program |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

[rtSelectorSetVisitProgram](#) was introduced in OptiX 1.0.

See also [rtSelectorGetVisitProgram](#), [rtProgramCreateFromPTXFile](#)

3.8.3.195 RTresult RTAPI rtSelectorValidate (RTselector *selector*)

Checks a Selector node for internal consistency.

Description

[rtSelectorValidate](#) recursively checks consistency of the Selector node *selector* and its children, i.e., it tries to validate the whole model sub-tree with *selector* as root. For a Selector node to be valid, it must be assigned a visit program, and the number of its children must match the number specified by [rtSelectorSetChildCount](#).

Parameters

| | | |
|----|-----------------|--|
| in | <i>selector</i> | Selector root node of a model sub-tree to be validated |
|----|-----------------|--|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtSelectorValidate](#) was introduced in OptiX 1.0.

See also [rtSelectorCreate](#), [rtSelectorDestroy](#), [rtSelectorGetContext](#), [rtSelectorSetVisitProgram](#), [rtSelectorSetChildCount](#), [rtSelectorSetChild](#)

3.8.3.196 RTresult RTAPI rtTextureSamplerCreate (RTcontext *context*, RTtexturesampler * *texturesampler*)

Creates a new texture sampler object.

Description

[rtTextureSamplerCreate](#) allocates a texture sampler object. Sets **texturesampler* to the handle of a newly created texture sampler within *context*. Returns [RT_ERROR_INVALID_VALUE](#) if *texturesampler* is *NULL*.

Parameters

| | | |
|-----|-----------------------|---|
| in | <i>context</i> | The context the texture sampler object will be created in |
| out | <i>texturesampler</i> | The return handle to the new texture sampler object |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTextureSamplerCreate](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerDestroy](#)

3.8.3.197 RTresult RTAPI rtTextureSamplerDestroy (RTtexturesampler *texturesampler*)

Destroys a texture sampler object.

Description

[rtTextureSamplerDestroy](#) removes *texturesampler* from its context and deletes it. *texturesampler* should be a value returned by [rtTextureSamplerCreate](#). After the call, *texturesampler* is no longer a valid handle. Any API object that referenced *texturesampler* will have its reference invalidated.

Parameters

| | | |
|----|-----------------------|--|
| in | <i>texturesampler</i> | Handle of the texture sampler to destroy |
|----|-----------------------|--|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTextureSamplerDestroy](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerCreate](#)

3.8.3.198 **RTresult RTAPI** [rtTextureSamplerGetArraySize](#) (**RTtexturesampler** *texturesampler*, unsigned int * *num_textures_in_array*)

Deprecated in OptiX 3.9. Use texture samplers with layered buffers instead. See [rtBufferCreate](#).

3.8.3.199 **RTresult RTAPI** [rtTextureSamplerGetBuffer](#) (**RTtexturesampler** *texturesampler*, unsigned int *deprecated0*, unsigned int *deprecated1*, **RTbuffer** * *buffer*)

Gets a buffer object handle from a texture sampler.

Description

[rtTextureSamplerGetBuffer](#) gets a buffer object from *texturesampler* and stores it in **buffer*.

Parameters

| | | |
|-----|-----------------------|---|
| in | <i>texturesampler</i> | The texture sampler object to be queried for the buffer |
| in | <i>deprecated0</i> | Deprecated in OptiX 3.9, must be 0 |
| in | <i>deprecated1</i> | Deprecated in OptiX 3.9, must be 0 |
| out | <i>buffer</i> | The return handle to the buffer attached to the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTextureSamplerGetBuffer](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerSetBuffer](#)

3.8.3.200 **RTresult RTAPI** [rtTextureSamplerGetContext](#) (**RTtexturesampler** *texturesampler*, **RTcontext** * *context*)

Gets the context object that created this texture sampler.

Description

[rtTextureSamplerGetContext](#) returns a handle to the context object that was used to create *texturesampler*. If *context* is *NULL*, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|-----------------------|---|
| in | <i>texturesampler</i> | The texture sampler object to be queried for its context |
| out | <i>context</i> | The return handle for the context object of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTextureSamplerGetContext](#) was introduced in OptiX 1.0.

See also [rtContextCreate](#)

3.8.3.201 **RTresult** RTAPI [rtTextureSamplerGetFilteringModes](#) (**RTtexturesampler** *texturesampler*, **RTfiltermode** * *minification*, **RTfiltermode** * *magnification*, **RTfiltermode** * *mipmapping*)

Gets the filtering modes of a texture sampler.

Description

[rtTextureSamplerGetFilteringModes](#) gets the minification, magnification and MIP mapping filtering modes from *texturesampler* and stores them in **minification*, **magnification* and **mipmapping*, respectively. See [rtTextureSamplerSetFilteringModes](#) for the values [RTfiltermode](#) may take.

Parameters

| | | |
|-----|-----------------------|---|
| in | <i>texturesampler</i> | The texture sampler object to be queried |
| out | <i>minification</i> | The return handle for the minification filtering mode of the texture sampler |
| out | <i>magnification</i> | The return handle for the magnification filtering mode of the texture sampler |
| out | <i>mipmapping</i> | The return handle for the MIP mapping filtering mode of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerGetFilteringModes](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerSetFilteringModes](#)

3.8.3.202 **RTresult** RTAPI [rtTextureSamplerGetId](#) (**RTtexturesampler** *texturesampler*, int * *texture_id*)

Returns the texture ID of this texture sampler.

Description

[rtTextureSamplerGetId](#) returns a handle to the texture sampler *texturesampler* to be used in OptiX programs on the device to reference the associated texture. The returned ID cannot be used on the host side. If *texture_id* is *NULL*, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|-----------------------|--|
| in | <i>texturesampler</i> | The texture sampler object to be queried for its ID |
| out | <i>texture_id</i> | The returned device-side texture ID of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerGetId](#) was introduced in OptiX 3.0.

See also [rtTextureSamplerCreate](#)

3.8.3.203 `RTresult RTAPI rtTextureSamplerGetIndexingMode (RTtexturesampler texturesampler, RTtextureindexmode * indexmode)`

Gets the indexing mode of a texture sampler.

Description

[rtTextureSamplerGetIndexingMode](#) gets the indexing mode of *texturesampler* and stores it in **indexmode*. See [rtTextureSamplerSetIndexingMode](#) for the values [RTtextureindexmode](#) may take.

Parameters

| | | |
|-----|-----------------------|--|
| in | <i>texturesampler</i> | The texture sampler object to be queried |
| out | <i>indexmode</i> | The return handle for the indexing mode of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerGetIndexingMode](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerSetIndexingMode](#)

3.8.3.204 `RTresult RTAPI rtTextureSamplerGetMaxAnisotropy (RTtexturesampler texturesampler, float * value)`

Gets the maximum anisotropy level for a texture sampler.

Description

[rtTextureSamplerGetMaxAnisotropy](#) gets the maximum anisotropy level for *texturesampler* and stores it in **value*.

Parameters

| | | |
|-----|-----------------------|---|
| in | <i>texturesampler</i> | The texture sampler object to be queried |
| out | <i>value</i> | The return handle for the maximum anisotropy level of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)

- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerGetMaxAnisotropy](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerSetMaxAnisotropy](#)

3.8.3.205 `RTresult RTAPI rtTextureSamplerGetMipLevelBias (RTtexturesampler texturesampler, float * value)`

Gets the mipmap offset for a texture sampler.

Description

[rtTextureSamplerGetMipLevelBias](#) gets the mipmap offset for *texturesampler* and stores it in **value*.

Parameters

| | | |
|-----|-----------------------|--|
| in | <i>texturesampler</i> | The texture sampler object to be queried |
| out | <i>value</i> | The return handle for the mipmap offset of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerGetMipLevelBias](#) was introduced in OptiX 3.9.

See also [rtTextureSamplerSetMipLevelBias](#)

3.8.3.206 `RTresult RTAPI rtTextureSamplerGetMipLevelClamp (RTtexturesampler texturesampler, float * minLevel, float * maxLevel)`

Gets the minimum and the maximum MIP level access range for a texture sampler.

Description

[rtTextureSamplerGetMipLevelClamp](#) gets the minimum and the maximum MIP level access range for *texturesampler* and stores it in **minLevel* and *maxLevel*.

Parameters

| | | |
|-----|-----------------------|---|
| in | <i>texturesampler</i> | The texture sampler object to be queried |
| out | <i>minLevel</i> | The return handle for the minimum mipmap level of the texture sampler |
| out | <i>maxLevel</i> | The return handle for the maximum mipmap level of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerGetMipLevelClamp](#) was introduced in OptiX 3.9.

See also [rtTextureSamplerSetMipLevelClamp](#)

3.8.3.207 **RTresult** RTAPI **rtTextureSamplerGetMipLevelCount** (**RTtexturesampler** *texturesampler*, unsigned int * *num_mip_levels*)

Deprecated in OptiX 3.9. Use [rtBufferGetMipLevelCount](#) instead.

3.8.3.208 **RTresult** RTAPI **rtTextureSamplerGetReadMode** (**RTtexturesampler** *texturesampler*, **RTtexturereadmode** * *readmode*)

Gets the read mode of a texture sampler.

Description

[rtTextureSamplerGetReadMode](#) gets the read mode of *texturesampler* and stores it in **readmode*. See [rtTextureSamplerSetReadMode](#) for a list of values [RTtexturereadmode](#) can take.

Parameters

| | | |
|-----|-----------------------|--|
| in | <i>texturesampler</i> | The texture sampler object to be queried |
| out | <i>readmode</i> | The return handle for the read mode of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerGetReadMode](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerSetReadMode](#)

3.8.3.209 **RTresult** RTAPI **rtTextureSamplerGetWrapMode** (**RTtexturesampler** *texturesampler*, unsigned int *dimension*, **RTwrapmode** * *wrapmode*)

Gets the wrap mode of a texture sampler.

Description

[rtTextureSamplerGetWrapMode](#) gets the texture wrapping mode of *texturesampler* and stores it in **wrapmode*. See [rtTextureSamplerSetWrapMode](#) for a list of values [RTwrapmode](#) can take.

Parameters

| | | |
|-----|-----------------------|--|
| in | <i>texturesampler</i> | The texture sampler object to be queried |
| in | <i>dimension</i> | Dimension for the wrapping |
| out | <i>wrapmode</i> | The return handle for the wrap mode of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerGetWrapMode](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerSetWrapMode](#)

3.8.3.210 **RTresult** RTAPI **rtTextureSamplerSetArraySize** (**RTtexturesampler** *texturesampler*, unsigned int *num_textures_in_array*)

Deprecated in OptiX 3.9. Use texture samplers with layered buffers instead. See [rtBufferCreate](#).

3.8.3.211 **RTresult** RTAPI **rtTextureSamplerSetBuffer** (**RTtexturesampler** *texturesampler*, unsigned int *deprecated0*, unsigned int *deprecated1*, **RTbuffer** *buffer*)

Attaches a buffer object to a texture sampler.

Description

[rtTextureSamplerSetBuffer](#) attaches *buffer* to *texturesampler*.

Parameters

| | | |
|----|-----------------------|---|
| in | <i>texturesampler</i> | The texture sampler object that will contain the buffer |
| in | <i>deprecated0</i> | Deprecated in OptiX 3.9, must be 0 |
| in | <i>deprecated1</i> | Deprecated in OptiX 3.9, must be 0 |
| in | <i>buffer</i> | The buffer to be attached to the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTextureSamplerSetBuffer](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerGetBuffer](#)

3.8.3.212 **RTresult** RTAPI **rtTextureSamplerSetFilteringModes** (**RTtexturesampler** *texturesampler*, **RTfiltermode** *minification*, **RTfiltermode** *magnification*, **RTfiltermode** *mipmapping*)

Sets the filtering modes of a texture sampler.

Description

[rtTextureSamplerSetFilteringModes](#) sets the minification, magnification and MIP mapping filter modes for *texture-sampler*. **RTfiltermode** must be one of the following values:

- [RT_FILTER_NEAREST](#)
- [RT_FILTER_LINEAR](#)
- [RT_FILTER_NONE](#)

These filter modes specify how the texture sampler will interpolate buffer data that has been attached to it. *minification* and *magnification* must be one of [RT_FILTER_NEAREST](#) or [RT_FILTER_LINEAR](#). *mipmapping* may be any of the three values but must be [RT_FILTER_NONE](#) if the texture sampler contains only a single MIP level or one of [RT_FILTER_NEAREST](#) or [RT_FILTER_LINEAR](#) if the texture sampler contains more than one MIP level.

Parameters

| | | |
|----|-----------------------|--|
| in | <i>texturesampler</i> | The texture sampler object to be changed |
| in | <i>minification</i> | The new minification filter mode of the texture sampler |
| in | <i>magnification</i> | The new magnification filter mode of the texture sampler |
| in | <i>mipmapping</i> | The new MIP mapping filter mode of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerSetFilteringModes](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerGetFilteringModes](#)

3.8.3.213 **RTresult** RTAPI **rtTextureSamplerSetIndexingMode** (**RTtexturesampler** *texturesampler*, **RTtextureindexmode** *indexmode*)

Sets whether texture coordinates for this texture sampler are normalized.

Description

[rtTextureSamplerSetIndexingMode](#) sets the indexing mode of *texturesampler* to *indexmode*. *indexmode* can take on one of the following values:

- [RT_TEXTURE_INDEX_NORMALIZED_COORDINATES](#),
- [RT_TEXTURE_INDEX_ARRAY_INDEX](#)

These values are used to control the interpretation of texture coordinates. If the index mode is set to [RT_TEXTURE_INDEX_NORMALIZED_COORDINATES](#), the texture is parameterized over [0,1]. If the index mode is set to [RT_TEXTURE_INDEX_ARRAY_INDEX](#) then texture coordinates are interpreted as array indices into the contents of the underlying buffer objects.

Parameters

| | | |
|----|-----------------------|--|
| in | <i>texturesampler</i> | The texture sampler object to be changed |
| in | <i>indexmode</i> | The new indexing mode of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerSetIndexingMode](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerGetIndexingMode](#)

3.8.3.214 **RTresult** RTAPI **rtTextureSamplerSetMaxAnisotropy** (**RTtexturesampler** *texturesampler*, **float** *value*)

Sets the maximum anisotropy of a texture sampler.

Description

[rtTextureSamplerSetMaxAnisotropy](#) sets the maximum anisotropy of *texturesampler* to *value*. A float value specifies the maximum anisotropy ratio to be used when doing anisotropic filtering. This value will be clamped to the range [1,16]

Parameters

| | | |
|----|-----------------------|---|
| in | <i>texturesampler</i> | The texture sampler object to be changed |
| in | <i>value</i> | The new maximum anisotropy level of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerSetMaxAnisotropy](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerGetMaxAnisotropy](#)

3.8.3.215 **RTresult** RTAPI [rtTextureSamplerSetMipLevelBias](#) (**RTtexturesampler** *texturesampler*, float *value*)

Sets the mipmap offset of a texture sampler.

Description

[rtTextureSamplerSetMipLevelBias](#) sets the offset to be applied to the calculated mipmap level.

Parameters

| | | |
|----|-----------------------|--|
| in | <i>texturesampler</i> | The texture sampler object to be changed |
| in | <i>value</i> | The new mipmap offset of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerSetMipLevelBias](#) was introduced in OptiX 3.9.

See also [rtTextureSamplerGetMipLevelBias](#)

3.8.3.216 **RTresult** RTAPI [rtTextureSamplerSetMipLevelClamp](#) (**RTtexturesampler** *texturesampler*, float *minLevel*, float *maxLevel*)

Sets the minimum and the maximum MIP level access range of a texture sampler.

Description

[rtTextureSamplerSetMipLevelClamp](#) sets lower end and the upper end of the MIP level range to clamp access to.

Parameters

| | | |
|----|-----------------------|---|
| in | <i>texturesampler</i> | The texture sampler object to be changed |
| in | <i>minLevel</i> | The new minimum mipmap level of the texture sampler |
| in | <i>maxLevel</i> | The new maximum mipmap level of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)

- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerSetMipLevelClamp](#) was introduced in OptiX 3.9.

See also [rtTextureSamplerGetMipLevelClamp](#)

3.8.3.217 `RTresult RTAPI rtTextureSamplerSetMipLevelCount (RTtexturesampler texturesampler, unsigned int num_mip_levels)`

Deprecated in OptiX 3.9. Use [rtBufferSetMipLevelCount](#) instead.

3.8.3.218 `RTresult RTAPI rtTextureSamplerSetReadMode (RTtexturesampler texturesampler, RTtexturereadmode readmode)`

Sets the read mode of a texture sampler.

Description

[rtTextureSamplerSetReadMode](#) sets the data read mode of *texturesampler* to *readmode*. *readmode* can take one of the following values:

- [RT_TEXTURE_READ_ELEMENT_TYPE](#)
- [RT_TEXTURE_READ_NORMALIZED_FLOAT](#)
- [RT_TEXTURE_READ_ELEMENT_TYPE_SRGB](#)
- [RT_TEXTURE_READ_NORMALIZED_FLOAT_SRGB](#)

[RT_TEXTURE_READ_ELEMENT_TYPE_SRGB](#) and [RT_TEXTURE_READ_NORMALIZED_FLOAT_SRGB](#) were introduced in OptiX 3.9 and apply sRGB to linear conversion during texture read for 8-bit integer buffer formats. *readmode* controls the returned value of the texture sampler when it is used to sample textures. [RT_TEXTURE_READ_ELEMENT_TYPE](#) will return data of the type of the underlying buffer objects. [RT_TEXTURE_READ_NORMALIZED_FLOAT](#) will return floating point values normalized by the range of the underlying type. If the underlying type is floating point, [RT_TEXTURE_READ_NORMALIZED_FLOAT](#) and [RT_TEXTURE_READ_ELEMENT_TYPE](#) are equivalent, always returning the unmodified floating point value.

For example, a texture sampler that samples a buffer of type [RT_FORMAT_UNSIGNED_BYTE](#) with a read mode of [RT_TEXTURE_READ_NORMALIZED_FLOAT](#) will convert integral values from the range [0,255] to floating point values in the range [0,1] automatically as the buffer is sampled from.

Parameters

| | | |
|----|-----------------------|--|
| in | <i>texturesampler</i> | The texture sampler object to be changed |
| in | <i>readmode</i> | The new read mode of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtTextureSamplerSetReadMode](#) was introduced in OptiX 1.0.

See also [rtTextureSamplerGetReadMode](#)

3.8.3.219 `RTresult RTAPI rtTextureSamplerSetWrapMode (RTtexturesampler texturesampler, unsigned int dimension, RTwrapmode wrapmode)`

Sets the wrapping mode of a texture sampler.

Description

`rtTextureSamplerSetWrapMode` sets the wrapping mode of *texturesampler* to *wrapmode* for the texture dimension specified by *dimension*. *wrapmode* can take one of the following values:

- [RT_WRAP_REPEAT](#)
- [RT_WRAP_CLAMP_TO_EDGE](#)
- [RT_WRAP_MIRROR](#)
- [RT_WRAP_CLAMP_TO_BORDER](#)

The wrapping mode controls the behavior of the texture sampler as texture coordinates wrap around the range specified by the indexing mode. These values mirror the CUDA behavior of textures. See CUDA programming guide for details.

Parameters

| | | |
|----|-----------------------|--|
| in | <i>texturesampler</i> | The texture sampler object to be changed |
| in | <i>dimension</i> | Dimension of the texture |
| in | <i>wrapmode</i> | The new wrap mode of the texture sampler |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

`rtTextureSamplerSetWrapMode` was introduced in OptiX 1.0. [RT_WRAP_MIRROR](#) and [RT_WRAP_CLAMP_TO_BORDER](#) were introduced in OptiX 3.0.

See also [rtTextureSamplerGetWrapMode](#)

3.8.3.220 `RTresult RTAPI rtTextureSamplerValidate (RTtexturesampler texturesampler)`

Validates the state of a texture sampler.

Description

`rtTextureSamplerValidate` checks *texturesampler* for completeness. If *texturesampler* does not have buffers attached to all of its MIP levels and array slices or if the filtering modes are incompatible with the current MIP level and array slice configuration then returns [RT_ERROR_INVALID_CONTEXT](#).

Parameters

| | | |
|----|-----------------------|-------------------------------------|
| in | <i>texturesampler</i> | The texture sampler to be validated |
|----|-----------------------|-------------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTextureSamplerValidate](#) was introduced in OptiX 1.0.

See also [rtContextValidate](#)

3.8.3.221 RTresult RTAPI rtTransformCreate (RTcontext *context*, RTtransform * *transform*)

Creates a new Transform node.

Description

Creates a new Transform node within the given context. For the node to be functional, a child node must be attached using [rtTransformSetChild](#). A transformation matrix can be associated with the transform node with [rtTransformSetMatrix](#). Sets **transform* to the handle of a newly created transform within *context*. Returns [RT_ERROR_INVALID_VALUE](#) if *transform* is *NULL*.

Parameters

| | | |
|-----|------------------|---|
| in | <i>context</i> | Specifies the rendering context of the Transform node |
| out | <i>transform</i> | New Transform node handle |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTransformCreate](#) was introduced in OptiX 1.0.

See also [rtTransformDestroy](#), [rtTransformValidate](#), [rtTransformGetContext](#), [rtTransformSetMatrix](#), [rtTransformGetMatrix](#), [rtTransformSetChild](#), [rtTransformGetChild](#), [rtTransformGetChildType](#)

3.8.3.222 RTresult RTAPI rtTransformDestroy (RTtransform *transform*)

Destroys a transform node.

Description

[rtTransformDestroy](#) removes *transform* from its context and deletes it. *transform* should be a value returned by [rtTransformCreate](#). No child graph nodes are destroyed. After the call, *transform* is no longer a valid handle.

Parameters

| | | |
|----|------------------|---|
| in | <i>transform</i> | Handle of the transform node to destroy |
|----|------------------|---|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTransformDestroy](#) was introduced in OptiX 1.0.

See also [rtTransformCreate](#), [rtTransformValidate](#), [rtTransformGetContext](#)

3.8.3.223 RTresult RTAPI rtTransformGetChild (RTtransform *transform*, RObject * *child*)

Returns the child node that is attached to a Transform node.

Description

[rtTransformGetChild](#) returns in *child* a handle of the child node currently attached to *transform*. The returned pointer is of generic type [RObject](#) and needs to be cast to the actual child type, which can be [RTgroup](#), [RTselector](#), [RTgeometrygroup](#), or [RTtransform](#). The actual type of *child* can be queried using [rtTransformGetChildType](#). Returns [RT_ERROR_INVALID_VALUE](#) if given a *NULL* pointer.

Parameters

| | | |
|-----|------------------|--|
| in | <i>transform</i> | Transform node handle |
| out | <i>child</i> | Child node handle. Can be { RTgroup , RTselector , RTgeometrygroup , RTtransform } |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTransformGetChild](#) was introduced in OptiX 1.0.

See also [rtTransformSetChild](#), [rtTransformGetChildType](#)

3.8.3.224 RTresult RTAPI rtTransformGetChildType (RTtransform *transform*, RObjecttype * *type*)

Returns type information about a Transform child node.

Description

[rtTransformGetChildType](#) queries the type of the child node attached to *transform*. If no child is attached, **type* is set to [RT_OBJECTTYPE_UNKNOWN](#) and [RT_ERROR_INVALID_VALUE](#) is returned. Returns [RT_ERROR_INVALID_VALUE](#) if given a *NULL* pointer. The returned type is one of:

- [RT_OBJECTTYPE_GROUP](#)
- [RT_OBJECTTYPE_GEOMETRY_GROUP](#)
- [RT_OBJECTTYPE_TRANSFORM](#)
- [RT_OBJECTTYPE_SELECTOR](#)

Parameters

| | | |
|-----|------------------|------------------------|
| in | <i>transform</i> | Transform node handle |
| out | <i>type</i> | Type of the child node |

Return values

Relevant return values:

- [RT_SUCCESS](#)

- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTransformGetChildType](#) was introduced in OptiX 1.0.

See also [rtTransformSetChild](#), [rtTransformGetChild](#)

3.8.3.225 **RTresult** RTAPI [rtTransformGetContext](#) (**RTtransform** *transform*, **RTcontext** * *context*)

Returns the context of a Transform node.

Description

[rtTransformGetContext](#) queries a transform node for its associated context. *transform* specifies the transform node to query, and should be a value returned by [rtTransformCreate](#). Sets **context* to the context associated with *transform*.

Parameters

| | | |
|-----|------------------|--|
| in | <i>transform</i> | Transform node handle |
| out | <i>context</i> | The context associated with <i>transform</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTransformGetContext](#) was introduced in OptiX 1.0.

See also [rtTransformCreate](#), [rtTransformDestroy](#), [rtTransformValidate](#)

3.8.3.226 **RTresult** RTAPI [rtTransformGetMatrix](#) (**RTtransform** *transform*, **int** *transpose*, **float** * *matrix*, **float** * *inverse_matrix*)

Returns the affine matrix and its inverse associated with a Transform node.

Description

[rtTransformGetMatrix](#) returns in *matrix* the affine matrix that is currently used to perform a transformation of the geometry contained in the sub-tree with *transform* as root. The corresponding inverse matrix will be returned in *inverse_matrix*. One or both pointers are allowed to be *NULL*. If *transpose* is 0, matrices are returned in row-major format, i.e., matrix rows are contiguously laid out in memory. If *transpose* is non-zero, matrices are returned in column-major format. If non-*NULL*, matrix pointers must point to a float array of at least 16 elements.

Parameters

| | | |
|-----|------------------|--|
| in | <i>transform</i> | Transform node handle |
| in | <i>transpose</i> | Flag indicating whether <i>matrix</i> and <i>inverse_matrix</i> should be transposed |
| out | <i>matrix</i> | Affine matrix (4x4 float array) |

| | | |
|-----|-----------------------|--------------------------------|
| out | <i>inverse_matrix</i> | Inverted form of <i>matrix</i> |
|-----|-----------------------|--------------------------------|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTransformGetMatrix](#) was introduced in OptiX 1.0.

See also [rtTransformSetMatrix](#)

3.8.3.227 RTresult RTAPI rtTransformSetChild (RTtransform *transform*, RObject *child*)

Attaches a child node to a Transform node.

Description

Attaches a child node *child* to the parent node *transform*. Legal child node types are [RTgroup](#), [RTselector](#), [RTgeometrygroup](#), and [RTtransform](#). A transform node must have exactly one child. If a transformation matrix has been attached to *transform* with [rtTransformSetMatrix](#), it is effective on the model sub-tree with *child* as root node.

Parameters

| | | |
|----|------------------|--|
| in | <i>transform</i> | Transform node handle |
| in | <i>child</i> | Child node to be attached. Can be { RTgroup , RTselector , RTgeometrygroup , RTtransform } |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTransformSetChild](#) was introduced in OptiX 1.0.

See also [rtTransformSetMatrix](#), [rtTransformGetChild](#), [rtTransformGetChildType](#)

3.8.3.228 RTresult RTAPI rtTransformSetMatrix (RTtransform *transform*, int *transpose*, const float * *matrix*, const float * *inverse_matrix*)

Associates an affine transformation matrix with a Transform node.

Description

[rtTransformSetMatrix](#) associates a 4x4 matrix with the Transform node *transform*. The provided transformation matrix results in a corresponding affine transformation of all geometry contained in the sub-tree with *transform* as root. At least one of the pointers *matrix* and *inverse_matrix* must be non-NULL. If exactly one pointer is valid, the other matrix will be computed. If both are valid, the matrices will be used as-is. If *transpose* is 0, source matrices are expected to be in row-major format, i.e., matrix rows are contiguously laid out in memory:

```
float matrix[4*4] = { a11, a12, a13, a14, a21, a22, a23, a24, a31, a32, a33, a34, a41, a42, a43, a44 };
```

Here, the translational elements a_{14} , a_{24} , and a_{34} are at the 4th, 8th, and 12th position the matrix array. If the supplied matrices are in column-major format, a non-0 *transpose* flag can be used to trigger an automatic transpose of the input matrices.

Parameters

| | | |
|----|-----------------------|--|
| in | <i>transform</i> | Transform node handle |
| in | <i>transpose</i> | Flag indicating whether <i>matrix</i> and <i>inverse_matrix</i> should be transposed |
| in | <i>matrix</i> | Affine matrix (4x4 float array) |
| in | <i>inverse_matrix</i> | Inverted form of <i>matrix</i> |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTransformSetMatrix](#) was introduced in OptiX 1.0.

See also [rtTransformGetMatrix](#)

3.8.3.229 RTresult RTAPI rtTransformValidate (RTtransform *transform*)

Checks a Transform node for internal consistency.

Description

[rtTransformValidate](#) recursively checks consistency of the Transform node *transform* and its child, i.e., it tries to validate the whole model sub-tree with *transform* as root. For a Transform node to be valid, it must have a child node attached. It is, however, not required to explicitly set a transformation matrix. Without a specified transformation matrix, the identity matrix is applied.

Parameters

| | | |
|----|------------------|---|
| in | <i>transform</i> | Transform root node of a model sub-tree to be validated |
|----|------------------|---|

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtTransformValidate](#) was introduced in OptiX 1.0.

See also [rtTransformCreate](#), [rtTransformDestroy](#), [rtTransformGetContext](#), [rtTransformSetMatrix](#), [rtTransformSetChild](#)

3.8.3.230 RTresult RTAPI rtVariableGet1f (RTvariable *v*, float * *f1*)

Functions designed to modify the value of a program variable.

Description

[rtVariableGet](#) functions return the value of a program variable or variable array. The target variable is specified by *v*.

The commands [rtVariableGet{1-2-3-4}{f-i-ui}v](#) are used to query the value of a program variable specified by *v* using the pointers passed as arguments as return locations for each component of the vector-typed variable. The number

specified in the command should match the number of components in the data type of the specified program variable (e.g., 1 for float, int, unsigned int; 2 for float2, int2, uint2, etc.). The suffix *f* indicates that floating-point values are expected to be returned, the suffix *i* indicates that integer values are expected, and the suffix *ui* indicates that unsigned integer values are expected, and this type should also match the data type of the specified program variable. The *f* variants of this function should be used to query values for program variables defined as float, float2, float3, float4, or arrays of these. The *i* variants of this function should be used to query values for program variables defined as int, int2, int3, int4, or arrays of these. The *ui* variants of this function should be used to query values for program variables defined as unsigned int, uint2, uint3, uint4, or arrays of these. The *v* variants of this function should be used to return the program variable's value to the array specified by parameter *v*. In this case, the array *v* should be large enough to accommodate all of the program variable's components.

The commands `rtVariableGetMatrix{2-3-4}x{2-3-4}fv` are used to query the value of a program variable whose data type is a matrix. The numbers in the command names are interpreted as the dimensionality of the matrix. For example, `2x4` indicates a 2 x 4 matrix with 2 columns and 4 rows (i.e., 8 values). If *transpose* is 0, the matrix is returned in row major order, otherwise in column major order.

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

`rtVariableGet` were introduced in OptiX 1.0.

See also `rtVariableSet`, [rtVariableGetType](#), [rtContextDeclareVariable](#)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>f1</i> | Float value to be returned |

3.8.3.231 RTresult RTAPI rtVariableGet1fv (RTvariable *v*, float * *f*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>f</i> | Array of float value(s) to be returned |

3.8.3.232 RTresult RTAPI rtVariableGet1i (RTvariable *v*, int * *i1*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>i1</i> | Integer value to be returned |

3.8.3.233 RTresult RTAPI rtVariableGet1iv (RTvariable *v*, int * *i*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>i</i> | Array of integer values to be returned |

3.8.3.234 RTresult RTAPI rtVariableGet1ui (RTvariable *v*, unsigned int * *u1*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>u1</i> | Unsigned integer value to be returned |

3.8.3.235 RTresult RTAPI rtVariableGet1uiv (RTvariable *v*, unsigned int * *u*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>u</i> | Array of unsigned integer values to be returned |

3.8.3.236 RTresult RTAPI rtVariableGet2f (RTvariable *v*, float * *f1*, float * *f2*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>f1</i> | Float value to be returned |
| in | <i>f2</i> | Float value to be returned |

3.8.3.237 RTresult RTAPI rtVariableGet2fv (RTvariable *v*, float * *f*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>f</i> | Array of float value(s) to be returned |

3.8.3.238 RTresult RTAPI rtVariableGet2i (RTvariable *v*, int * *i1*, int * *i2*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>i1</i> | Integer value to be returned |
| in | <i>i2</i> | Integer value to be returned |

3.8.3.239 RTresult RTAPI rtVariableGet2iv (RTvariable *v*, int * *i*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>i</i> | Array of integer values to be returned |

3.8.3.240 RTresult RTAPI rtVariableGet2ui (RTvariable *v*, unsigned int * *u1*, unsigned int * *u2*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>u1</i> | Unsigned integer value to be returned |
| in | <i>u2</i> | Unsigned integer value to be returned |

3.8.3.241 RTresult RTAPI rtVariableGet2uiv (RTvariable *v*, unsigned int * *u*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>u</i> | Array of unsigned integer values to be returned |

3.8.3.242 RTresult RTAPI rtVariableGet3f (RTvariable *v*, float * *f1*, float * *f2*, float * *f3*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>f1</i> | Float value to be returned |
| in | <i>f2</i> | Float value to be returned |
| in | <i>f3</i> | Float value to be returned |

3.8.3.243 RTresult RTAPI rtVariableGet3fv (RTvariable *v*, float * *f*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>f</i> | Array of float value(s) to be returned |

3.8.3.244 RTresult RTAPI rtVariableGet3i (RTvariable *v*, int * *i1*, int * *i2*, int * *i3*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>i1</i> | Integer value to be returned |
| in | <i>i2</i> | Integer value to be returned |
| in | <i>i3</i> | Integer value to be returned |

3.8.3.245 RTresult RTAPI rtVariableGet3iv (RTvariable *v*, int * *i*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>i</i> | Array of integer values to be returned |

3.8.3.246 RTresult RTAPI rtVariableGet3ui (RTvariable *v*, unsigned int * *u1*, unsigned int * *u2*, unsigned int * *u3*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>u1</i> | Unsigned integer value to be returned |
| in | <i>u2</i> | Unsigned integer value to be returned |
| in | <i>u3</i> | Unsigned integer value to be returned |

3.8.3.247 RTresult RTAPI rtVariableGet3uiv (RTvariable *v*, unsigned int * *u*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>u</i> | Array of unsigned integer values to be returned |

3.8.3.248 RTresult RTAPI rtVariableGet4f (RTvariable *v*, float * *f1*, float * *f2*, float * *f3*, float * *f4*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>f1</i> | Float value to be returned |
| in | <i>f2</i> | Float value to be returned |
| in | <i>f3</i> | Float value to be returned |
| in | <i>f4</i> | Float value to be returned |

3.8.3.249 RTresult RTAPI rtVariableGet4fv (RTvariable *v*, float * *f*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>f</i> | Array of float value(s) to be returned |

3.8.3.250 RTresult RTAPI rtVariableGet4i (RTvariable *v*, int * *i1*, int * *i2*, int * *i3*, int * *i4*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>i1</i> | Integer value to be returned |
| in | <i>i2</i> | Integer value to be returned |
| in | <i>i3</i> | Integer value to be returned |
| in | <i>i4</i> | Integer value to be returned |

3.8.3.251 RTresult RTAPI rtVariableGet4iv (RTvariable *v*, int * *i*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>i</i> | Array of integer values to be returned |

3.8.3.252 RTresult RTAPI rtVariableGet4ui (RTvariable *v*, unsigned int * *u1*, unsigned int * *u2*, unsigned int * *u3*, unsigned int * *u4*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>u1</i> | Unsigned integer value to be returned |
| in | <i>u2</i> | Unsigned integer value to be returned |
| in | <i>u3</i> | Unsigned integer value to be returned |
| in | <i>u4</i> | Unsigned integer value to be returned |

3.8.3.253 RTresult RTAPI rtVariableGet4uiv (RTvariable *v*, unsigned int * *u*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>u</i> | Array of unsigned integer values to be returned |

3.8.3.254 RTresult RTAPI rtVariableGetAnnotation (RTvariable *v*, const char ** *annotation_return*)

Queries the annotation string of a program variable.

Description

[rtVariableGetAnnotation](#) queries a program variable's annotation string. A pointer to the string containing the annotation is returned in **annotation_return*. If *v* is not a valid variable, this call sets **annotation_return* to *NULL* and returns [RT_ERROR_INVALID_VALUE](#). **annotation_return* will point to valid memory until another API function that

returns a string is called.

Parameters

| | | |
|-----|---------------------------------------|--|
| in | <i>v</i> | Specifies the program variable to be queried |
| out | <i>annotation_</i> - <i>return</i> | Returns the program variable's annotation string |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtVariableGetAnnotation](#) was introduced in OptiX 1.0.

See also [rtDeclareVariable](#), [rtDeclareAnnotation](#)

3.8.3.255 **RTresult** RTAPI [rtVariableGetContext](#) (**RTvariable** *v*, **RTcontext** * *context*)

Returns the context associated with a program variable.

Description

[rtVariableGetContext](#) queries the context associated with a program variable. The target variable is specified by *v*. The context of the program variable is returned to **context* if the pointer *context* is not *NULL*. If *v* is not a valid variable, **context* is set to *NULL* and [RT_ERROR_INVALID_VALUE](#) is returned.

Parameters

| | | |
|-----|----------------|--|
| in | <i>v</i> | Specifies the program variable to be queried |
| out | <i>context</i> | Returns the context associated with the program variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtVariableGetContext](#) was introduced in OptiX 1.0.

See also [rtContextDeclareVariable](#)

3.8.3.256 **RTresult** RTAPI [rtVariableGetMatrix2x2fv](#) (**RTvariable** *v*, int *transpose*, float * *m*)

Parameters

| | | |
|----|------------------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>transpose</i> | Specify(ies) row-major or column-major order |
| in | <i>m</i> | Array of float values to be returned |

3.8.3.257 **RTresult** RTAPI [rtVariableGetMatrix2x3fv](#) (**RTvariable** *v*, int *transpose*, float * *m*)

Parameters

| | | |
|----|------------------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>transpose</i> | Specify(ies) row-major or column-major order |
| in | <i>m</i> | Array of float values to be returned |

3.8.3.258 RTresult RTAPI rtVariableGetMatrix2x4fv (RTvariable *v*, int *transpose*, float * *m*)

Parameters

| | | |
|----|------------------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>transpose</i> | Specify(ies) row-major or column-major order |
| in | <i>m</i> | Array of float values to be returned |

3.8.3.259 RTresult RTAPI rtVariableGetMatrix3x2fv (RTvariable *v*, int *transpose*, float * *m*)

Parameters

| | | |
|----|------------------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>transpose</i> | Specify(ies) row-major or column-major order |
| in | <i>m</i> | Array of float values to be returned |

3.8.3.260 RTresult RTAPI rtVariableGetMatrix3x3fv (RTvariable *v*, int *transpose*, float * *m*)

Parameters

| | | |
|----|------------------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>transpose</i> | Specify(ies) row-major or column-major order |
| in | <i>m</i> | Array of float values to be returned |

3.8.3.261 RTresult RTAPI rtVariableGetMatrix3x4fv (RTvariable *v*, int *transpose*, float * *m*)

Parameters

| | | |
|----|------------------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>transpose</i> | Specify(ies) row-major or column-major order |
| in | <i>m</i> | Array of float values to be returned |

3.8.3.262 RTresult RTAPI rtVariableGetMatrix4x2fv (RTvariable *v*, int *transpose*, float * *m*)

Parameters

| | | |
|----|------------------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>transpose</i> | Specify(ies) row-major or column-major order |
| in | <i>m</i> | Array of float values to be returned |

3.8.3.263 RTresult RTAPI rtVariableGetMatrix4x3fv (RTvariable *v*, int *transpose*, float * *m*)

Parameters

| | | |
|----|------------------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>transpose</i> | Specify(ies) row-major or column-major order |
| in | <i>m</i> | Array of float values to be returned |

3.8.3.264 RTresult RTAPI rtVariableGetMatrix4x4fv (RTvariable *v*, int *transpose*, float * *m*)

Parameters

| | | |
|----|------------------|--|
| in | <i>v</i> | Specifies the program variable whose value is to be returned |
| in | <i>transpose</i> | Specify(ies) row-major or column-major order |
| in | <i>m</i> | Array of float values to be returned |

3.8.3.265 RTresult RTAPI rtVariableGetName (RTvariable *v*, const char ** *name_return*)

Queries the name of a program variable.

Description

Queries a program variable's name. The variable of interest is specified by *variable*, which should be a value returned by [rtContextDeclareVariable](#). A pointer to the string containing the name of the variable is returned in **name_return*. If *v* is not a valid variable, this call sets **name_return* to *NULL* and returns [RT_ERROR_INVALID_VALUE](#). **name_return* will point to valid memory until another API function that returns a string is called.

Parameters

| | | |
|-----|--------------------|--|
| in | <i>v</i> | Specifies the program variable to be queried |
| out | <i>name_return</i> | Returns the program variable's name |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)

History

[rtVariableGetName](#) was introduced in OptiX 1.0.

See also [rtContextDeclareVariable](#)

3.8.3.266 RTresult RTAPI rtVariableGetObject (RTvariable *v*, RObject * *object*)

Returns the value of a OptiX object program variable.

Description

[rtVariableGetObject](#) queries the value of a program variable whose data type is a OptiX object. The target variable is specified by *v*. The value of the program variable is returned in **object*. The concrete type of the program variable can be queried using [rtVariableGetType](#), and the [RObject](#) handle returned by [rtVariableGetObject](#) may safely be cast to an OptiX handle of corresponding type. If *v* is not a valid variable, this call sets **object* to *NULL* and returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|---------------|--|
| in | <i>v</i> | Specifies the program variable to be queried |
| out | <i>object</i> | Returns the value of the program variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

[rtVariableGetObject](#) was introduced in OptiX 1.0.

See also [rtVariableSetObject](#), [rtVariableGetType](#), [rtContextDeclareVariable](#)

3.8.3.267 RTresult RTAPI rtVariableGetSize (RTvariable v, RTsize * size)

Queries the size, in bytes, of a variable.

Description

[rtVariableGetSize](#) queries a declared program variable for its size in bytes. This is most often used to query the size of a variable that has a user-defined type. Builtin types (int, float, unsigned int, etc.) may be queried, but object typed variables, such as buffers, texture samplers and graph nodes, cannot be queried and will return [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|------|--|
| in | v | Specifies the program variable to be queried |
| out | size | Specifies a pointer where the size of the variable, in bytes, will be returned |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtVariableGetSize](#) was introduced in OptiX 1.0.

See also [rtVariableGetUserData](#), [rtContextDeclareVariable](#)

3.8.3.268 RTresult RTAPI rtVariableGetType (RTvariable v, RTOBJECTTYPE * type_return)

Returns type information about a program variable.

Description

[rtVariableGetType](#) queries a program variable's type. The variable of interest is specified by v. The program variable's type enumeration is returned in **type_return*, if it is not *NULL*. It is one of the following:

- [RT_OBJECTTYPE_UNKNOWN](#)
- [RT_OBJECTTYPE_GROUP](#)
- [RT_OBJECTTYPE_GEOMETRY_GROUP](#)
- [RT_OBJECTTYPE_TRANSFORM](#)
- [RT_OBJECTTYPE_SELECTOR](#)
- [RT_OBJECTTYPE_GEOMETRY_INSTANCE](#)
- [RT_OBJECTTYPE_BUFFER](#)
- [RT_OBJECTTYPE_TEXTURE_SAMPLER](#)
- [RT_OBJECTTYPE_OBJECT](#)
- [RT_OBJECTTYPE_MATRIX_FLOAT2x2](#)
- [RT_OBJECTTYPE_MATRIX_FLOAT2x3](#)
- [RT_OBJECTTYPE_MATRIX_FLOAT2x4](#)
- [RT_OBJECTTYPE_MATRIX_FLOAT3x2](#)

- [RT_OBJECTTYPE_MATRIX_FLOAT3x3](#)
- [RT_OBJECTTYPE_MATRIX_FLOAT3x4](#)
- [RT_OBJECTTYPE_MATRIX_FLOAT4x2](#)
- [RT_OBJECTTYPE_MATRIX_FLOAT4x3](#)
- [RT_OBJECTTYPE_MATRIX_FLOAT4x4](#)
- [RT_OBJECTTYPE_FLOAT](#)
- [RT_OBJECTTYPE_FLOAT2](#)
- [RT_OBJECTTYPE_FLOAT3](#)
- [RT_OBJECTTYPE_FLOAT4](#)
- [RT_OBJECTTYPE_INT](#)
- [RT_OBJECTTYPE_INT2](#)
- [RT_OBJECTTYPE_INT3](#)
- [RT_OBJECTTYPE_INT4](#)
- [RT_OBJECTTYPE_UNSIGNED_INT](#)
- [RT_OBJECTTYPE_UNSIGNED_INT2](#)
- [RT_OBJECTTYPE_UNSIGNED_INT3](#)
- [RT_OBJECTTYPE_UNSIGNED_INT4](#)
- [RT_OBJECTTYPE_USER](#)

Sets **type_return* to [RT_OBJECTTYPE_UNKNOWN](#) if *v* is not a valid variable. Returns [RT_ERROR_INVALID_VALUE](#) if given a *NULL* pointer.

Parameters

| | | |
|-----|--------------------|--|
| in | <i>v</i> | Specifies the program variable to be queried |
| out | <i>type_return</i> | Returns the type of the program variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtVariableGetType](#) was introduced in OptiX 1.0.

See also [rtContextDeclareVariable](#)

3.8.3.269 **RTresult** RTAPI [rtVariableGetUserData](#) (**RTvariable** *v*, **RTsize** *size*, **void ****ptr*)

Defined.

Description

[rtVariableGetUserData](#) queries the value of a program variable whose data type is user-defined. The variable of interest is specified by *v*. The size of the variable's value must match the value given by the parameter *size*. The value of the program variable is copied to the memory region pointed to by *ptr*. The storage at location *ptr* must be large enough to accommodate all of the program variable's value data. If *v* is not a valid variable, this call has no effect and returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|-----|-------------|--|
| in | <i>v</i> | Specifies the program variable to be queried |
| in | <i>size</i> | Specifies the size of the program variable, in bytes |
| out | <i>ptr</i> | Location in which to store the value of the variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

[rtVariableGetUserData](#) was introduced in OptiX 1.0.

See also [rtVariableSetUserData](#), [rtContextDeclareVariable](#)

3.8.3.270 RTresult RTAPI rtVariableSet1f (RTvariable *v*, float *f1*)

Functions designed to modify the value of a program variable.

Description

`rtVariableSet` functions modify the value of a program variable or variable array. The target variable is specified by *v*, which should be a value returned by [rtContextGetVariable](#).

The commands `rtVariableSet{1-2-3-4}{f-i-ui}v` are used to modify the value of a program variable specified by *v* using the values passed as arguments. The number specified in the command should match the number of components in the data type of the specified program variable (e.g., 1 for float, int, unsigned int; 2 for float2, int2, uint2, etc.). The suffix *f* indicates that *v* has floating point type, the suffix *i* indicates that *v* has integral type, and the suffix *ui* indicates that *v* has unsigned integral type. The *v* variants of this function should be used to load the program variable's value from the array specified by parameter *v*. In this case, the array *v* should contain as many elements as there are program variable components.

The commands `rtVariableSetMatrix{2-3-4}x{2-3-4}fv` are used to modify the value of a program variable whose data type is a matrix. The numbers in the command names are the number of rows and columns, respectively. For example, `2x4` indicates a matrix with 2 rows and 4 columns (i.e., 8 values). If *transpose* is 0, the matrix is specified in row-major order, otherwise in column-major order or, equivalently, as a matrix with the number of rows and columns swapped in row-major order.

If *v* is not a valid variable, these calls have no effect and return [RT_ERROR_INVALID_VALUE](#)

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)

History

`rtVariableSet` were introduced in OptiX 1.0.

See also [rtVariableGet](#), [rtVariableSet](#), [rtDeclareVariable](#)

Parameters

| | | |
|----|-----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>f1</i> | Specifies the new float value of the program variable |

3.8.3.271 RTresult RTAPI rtVariableSet1fv (RTvariable *v*, const float * *f*)

Parameters

| | | |
|----|----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>f</i> | Array of float values to set the variable to |

3.8.3.272 RTresult RTAPI rtVariableSet1i (RTvariable *v*, int *i1*)

Parameters

| | | |
|----|-----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>i1</i> | Specifies the new integer value of the program variable |

3.8.3.273 RTresult RTAPI rtVariableSet1iv (RTvariable *v*, const int * *i*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>i</i> | Array of integer values to set the variable to |

3.8.3.274 RTresult RTAPI rtVariableSet1ui (RTvariable *v*, unsigned int *u1*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>u1</i> | Specifies the new unsigned integer value of the program variable |

3.8.3.275 RTresult RTAPI rtVariableSet1uiv (RTvariable *v*, const unsigned int * *u*)

Parameters

| | | |
|----|----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>u</i> | Array of unsigned integer values to set the variable to |

3.8.3.276 RTresult RTAPI rtVariableSet2f (RTvariable *v*, float *f1*, float *f2*)

Parameters

| | | |
|----|-----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>f1</i> | Specifies the new float value of the program variable |
| in | <i>f2</i> | Specifies the new float value of the program variable |

3.8.3.277 RTresult RTAPI rtVariableSet2fv (RTvariable *v*, const float * *f*)

Parameters

| | | |
|----|----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>f</i> | Array of float values to set the variable to |

3.8.3.278 RTresult RTAPI rtVariableSet2i (RTvariable *v*, int *i1*, int *i2*)

Parameters

| | | |
|----|-----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>i1</i> | Specifies the new integer value of the program variable |
| in | <i>i2</i> | Specifies the new integer value of the program variable |

3.8.3.279 RTresult RTAPI rtVariableSet2iv (RTvariable *v*, const int * *i*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>i</i> | Array of integer values to set the variable to |

3.8.3.280 RTresult RTAPI rtVariableSet2ui (RTvariable *v*, unsigned int *u1*, unsigned int *u2*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>u1</i> | Specifies the new unsigned integer value of the program variable |
| in | <i>u2</i> | Specifies the new unsigned integer value of the program variable |

3.8.3.281 RTresult RTAPI rtVariableSet2uiv (RTvariable *v*, const unsigned int * *u*)

Parameters

| | | |
|----|----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>u</i> | Array of unsigned integer values to set the variable to |

3.8.3.282 RTresult RTAPI rtVariableSet3f (RTvariable *v*, float *f1*, float *f2*, float *f3*)

Parameters

| | | |
|----|-----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>f1</i> | Specifies the new float value of the program variable |
| in | <i>f2</i> | Specifies the new float value of the program variable |
| in | <i>f3</i> | Specifies the new float value of the program variable |

3.8.3.283 RTresult RTAPI rtVariableSet3fv (RTvariable *v*, const float * *f*)

Parameters

| | | |
|----|----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>f</i> | Array of float values to set the variable to |

3.8.3.284 RTresult RTAPI rtVariableSet3i (RTvariable *v*, int *i1*, int *i2*, int *i3*)

Parameters

| | | |
|----|-----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>i1</i> | Specifies the new integer value of the program variable |
| in | <i>i2</i> | Specifies the new integer value of the program variable |
| in | <i>i3</i> | Specifies the new integer value of the program variable |

3.8.3.285 RTresult RTAPI rtVariableSet3iv (RTvariable *v*, const int * *i*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>i</i> | Array of integer values to set the variable to |

3.8.3.286 RTresult RTAPI rtVariableSet3ui (RTvariable *v*, unsigned int *u1*, unsigned int *u2*, unsigned int *u3*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>u1</i> | Specifies the new unsigned integer value of the program variable |
| in | <i>u2</i> | Specifies the new unsigned integer value of the program variable |
| in | <i>u3</i> | Specifies the new unsigned integer value of the program variable |

3.8.3.287 RTresult RTAPI rtVariableSet3uiv (RTvariable *v*, const unsigned int * *u*)

Parameters

| | | |
|----|----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>u</i> | Array of unsigned integer values to set the variable to |

3.8.3.288 RTresult RTAPI rtVariableSet4f (RTvariable *v*, float *f1*, float *f2*, float *f3*, float *f4*)

Parameters

| | | |
|----|-----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>f1</i> | Specifies the new float value of the program variable |
| in | <i>f2</i> | Specifies the new float value of the program variable |
| in | <i>f3</i> | Specifies the new float value of the program variable |
| in | <i>f4</i> | Specifies the new float value of the program variable |

3.8.3.289 RTresult RTAPI rtVariableSet4fv (RTvariable *v*, const float * *f*)

Parameters

| | | |
|----|----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>f</i> | Array of float values to set the variable to |

3.8.3.290 RTresult RTAPI rtVariableSet4i (RTvariable *v*, int *i1*, int *i2*, int *i3*, int *i4*)

Parameters

| | | |
|----|-----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>i1</i> | Specifies the new integer value of the program variable |
| in | <i>i2</i> | Specifies the new integer value of the program variable |
| in | <i>i3</i> | Specifies the new integer value of the program variable |
| in | <i>i4</i> | Specifies the new integer value of the program variable |

3.8.3.291 RTresult RTAPI rtVariableSet4iv (RTvariable *v*, const int * *i*)

Parameters

| | | |
|----|----------|--|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>i</i> | Array of integer values to set the variable to |

3.8.3.292 RTresult RTAPI rtVariableSet4ui (RTvariable *v*, unsigned int *u1*, unsigned int *u2*, unsigned int *u3*, unsigned int *u4*)

Parameters

| | | |
|----|-----------|--|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>u1</i> | Specifies the new unsigned integer value of the program variable |
| in | <i>u2</i> | Specifies the new unsigned integer value of the program variable |
| in | <i>u3</i> | Specifies the new unsigned integer value of the program variable |
| in | <i>u4</i> | Specifies the new unsigned integer value of the program variable |

3.8.3.293 RTresult RTAPI rtVariableSet4uiv (RTvariable *v*, const unsigned int * *u*)

Parameters

| | | |
|----|----------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>u</i> | Array of unsigned integer values to set the variable to |

3.8.3.294 RTresult RTAPI rtVariableSetMatrix2x2fv (RTvariable *v*, int *transpose*, const float * *m*)

Parameters

| | | |
|----|------------------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>transpose</i> | Specifies row-major or column-major order |
| in | <i>m</i> | Array of float values to set the matrix to |

3.8.3.295 RTresult RTAPI rtVariableSetMatrix2x3fv (RTvariable *v*, int *transpose*, const float * *m*)

Parameters

| | | |
|----|------------------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>transpose</i> | Specifies row-major or column-major order |
| in | <i>m</i> | Array of float values to set the matrix to |

3.8.3.296 RTresult RTAPI rtVariableSetMatrix2x4fv (RTvariable *v*, int *transpose*, const float * *m*)

Parameters

| | | |
|----|------------------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>transpose</i> | Specifies row-major or column-major order |
| in | <i>m</i> | Array of float values to set the matrix to |

3.8.3.297 RTresult RTAPI rtVariableSetMatrix3x2fv (RTvariable *v*, int *transpose*, const float * *m*)

Parameters

| | | |
|----|------------------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>transpose</i> | Specifies row-major or column-major order |
| in | <i>m</i> | Array of float values to set the matrix to |

3.8.3.298 RTresult RTAPI rtVariableSetMatrix3x3fv (RTvariable *v*, int *transpose*, const float * *m*)

Parameters

| | | |
|----|------------------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>transpose</i> | Specifies row-major or column-major order |
| in | <i>m</i> | Array of float values to set the matrix to |

3.8.3.299 RTresult RTAPI rtVariableSetMatrix3x4fv (RTvariable *v*, int *transpose*, const float * *m*)

Parameters

| | | |
|----|------------------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>transpose</i> | Specifies row-major or column-major order |
| in | <i>m</i> | Array of float values to set the matrix to |

3.8.3.300 `RTresult RTAPI rtVariableSetMatrix4x2fv (RTvariable v, int transpose, const float * m)`

Parameters

| | | |
|----|------------------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>transpose</i> | Specifies row-major or column-major order |
| in | <i>m</i> | Array of float values to set the matrix to |

3.8.3.301 `RTresult RTAPI rtVariableSetMatrix4x3fv (RTvariable v, int transpose, const float * m)`

Parameters

| | | |
|----|------------------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>transpose</i> | Specifies row-major or column-major order |
| in | <i>m</i> | Array of float values to set the matrix to |

3.8.3.302 `RTresult RTAPI rtVariableSetMatrix4x4fv (RTvariable v, int transpose, const float * m)`

Parameters

| | | |
|----|------------------|---|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>transpose</i> | Specifies row-major or column-major order |
| in | <i>m</i> | Array of float values to set the matrix to |

3.8.3.303 `RTresult RTAPI rtVariableSetObject (RTvariable v, RObject object)`

Sets a program variable value to a OptiX object.

Description

`rtVariableSetObject` sets a program variable to an OptiX object value. The target variable is specified by *v*. The new value of the program variable is specified by *object*. The concrete type of *object* can be one of [RTbuffer](#), [RTtexturesampler](#), [RTgroup](#), [RTprogram](#), [RTselector](#), [RTgeometrygroup](#), or [RTtransform](#). If *v* is not a valid variable or *object* is not a valid OptiX object, this call has no effect and returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|----|---------------|---|
| in | <i>v</i> | Specifies the program variable to be set |
| in | <i>object</i> | Specifies the new value of the program variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

`rtVariableSetObject` was introduced in OptiX 1.0. The ability to bind an [RTprogram](#) to a variable was introduced in OptiX 3.0.

See also [rtVariableGetObject](#), [rtContextDeclareVariable](#)

3.8.3.304 RTresult RTAPI rtVariableSetUserData (RTvariable *v*, RTsize *size*, const void * *ptr*)

Defined.

Description

[rtVariableSetUserData](#) modifies the value of a program variable whose data type is user-defined. The value copied into the variable is defined by an arbitrary region of memory, pointed to by *ptr*. The size of the memory region is given by *size*. The target variable is specified by *v*. If *v* is not a valid variable, this call has no effect and returns [RT_ERROR_INVALID_VALUE](#).

Parameters

| | | |
|----|-------------|--|
| in | <i>v</i> | Specifies the program variable to be modified |
| in | <i>size</i> | Specifies the size of the new value, in bytes |
| in | <i>ptr</i> | Specifies a pointer to the new value of the program variable |

Return values

Relevant return values:

- [RT_SUCCESS](#)
- [RT_ERROR_INVALID_CONTEXT](#)
- [RT_ERROR_INVALID_VALUE](#)
- [RT_ERROR_MEMORY_ALLOCATION_FAILED](#)
- [RT_ERROR_TYPE_MISMATCH](#)

History

[rtVariableSetUserData](#) was introduced in OptiX 1.0.

See also [rtVariableGetUserData](#), [rtContextDeclareVariable](#)

3.9 optix_prime.h File Reference

3.9.1 Detailed Description

OptiX Prime public API.

Author

NVIDIA Corporation OptiX Prime public API

Macros

- `#define OPTIX_PRIME_VERSION`
- `#define RTPAPI __declspec(dllimport)`

Typedefs

- `typedef unsigned int RTPsize`
- `typedef struct RTPcontext_api * RTPcontext`
- `typedef struct RTPmodel_api * RTPmodel`
- `typedef struct RTPquery_api * RTPquery`
- `typedef struct RTPbufferdesc_api * RTPbufferdesc`
- `typedef struct CUstream_st * cudaStream_t`

Functions

- [RTPresult](#) RTPAPI [rtpContextCreate](#) ([RTPcontexttype](#) type, [RTPcontext](#) *context)
- [RTPresult](#) RTPAPI [rtpContextSetCudaDeviceNumbers](#) ([RTPcontext](#) context, unsigned deviceCount, const unsigned *deviceNumbers)
- [RTPresult](#) RTPAPI [rtpContextSetCpuThreads](#) ([RTPcontext](#) context, unsigned numThreads)
- [RTPresult](#) RTPAPI [rtpContextDestroy](#) ([RTPcontext](#) context)
- [RTPresult](#) RTPAPI [rtpContextGetLastErrorString](#) ([RTPcontext](#) context, const char **return_string)
- [RTPresult](#) RTPAPI [rtpBufferDescCreate](#) ([RTPcontext](#) context, [RTPbufferformat](#) format, [RTPbuffertype](#) type, void *buffer, [RTPbufferdesc](#) *desc)
- [RTPresult](#) RTPAPI [rtpBufferDescGetContext](#) ([RTPbufferdesc](#) desc, [RTPcontext](#) *context)
- [RTPresult](#) RTPAPI [rtpBufferDescSetRange](#) ([RTPbufferdesc](#) desc, RTPsize begin, RTPsize end)
- [RTPresult](#) RTPAPI [rtpBufferDescSetStride](#) ([RTPbufferdesc](#) desc, unsigned strideBytes)
- [RTPresult](#) RTPAPI [rtpBufferDescSetCudaDeviceNumber](#) ([RTPbufferdesc](#) desc, unsigned deviceNumber)
- [RTPresult](#) RTPAPI [rtpBufferDescDestroy](#) ([RTPbufferdesc](#) desc)
- [RTPresult](#) RTPAPI [rtpModelCreate](#) ([RTPcontext](#) context, [RTPmodel](#) *model)
- [RTPresult](#) RTPAPI [rtpModelGetContext](#) ([RTPmodel](#) model, [RTPcontext](#) *context)
- [RTPresult](#) RTPAPI [rtpModelSetTriangles](#) ([RTPmodel](#) model, [RTPbufferdesc](#) indices, [RTPbufferdesc](#) vertices)
- [RTPresult](#) RTPAPI [rtpModelSetInstances](#) ([RTPmodel](#) model, [RTPbufferdesc](#) instances, [RTPbufferdesc](#) transforms)
- [RTPresult](#) RTPAPI [rtpModelUpdate](#) ([RTPmodel](#) model, unsigned hints)
- [RTPresult](#) RTPAPI [rtpModelFinish](#) ([RTPmodel](#) model)
- [RTPresult](#) RTPAPI [rtpModelGetFinished](#) ([RTPmodel](#) model, int *isFinished)
- [RTPresult](#) RTPAPI [rtpModelCopy](#) ([RTPmodel](#) model, [RTPmodel](#) srcModel)
- [RTPresult](#) RTPAPI [rtpModelSetBuilderParameter](#) ([RTPmodel](#) model_api, [RTPbuilderparam](#) param, RTPsize size, const void *ptr)
- [RTPresult](#) RTPAPI [rtpModelDestroy](#) ([RTPmodel](#) model)
- [RTPresult](#) RTPAPI [rtpQueryCreate](#) ([RTPmodel](#) model, [RTPquerytype](#) queryType, [RTPquery](#) *query)
- [RTPresult](#) RTPAPI [rtpQueryGetContext](#) ([RTPquery](#) query, [RTPcontext](#) *context)
- [RTPresult](#) RTPAPI [rtpQuerySetRays](#) ([RTPquery](#) query, [RTPbufferdesc](#) rays)
- [RTPresult](#) RTPAPI [rtpQuerySetHits](#) ([RTPquery](#) query, [RTPbufferdesc](#) hits)
- [RTPresult](#) RTPAPI [rtpQueryExecute](#) ([RTPquery](#) query, unsigned hints)
- [RTPresult](#) RTPAPI [rtpQueryFinish](#) ([RTPquery](#) query)
- [RTPresult](#) RTPAPI [rtpQueryGetFinished](#) ([RTPquery](#) query, int *isFinished)
- [RTPresult](#) RTPAPI [rtpQuerySetCudaStream](#) ([RTPquery](#) query, cudaStream_t stream)
- [RTPresult](#) RTPAPI [rtpQueryDestroy](#) ([RTPquery](#) query)
- [RTPresult](#) RTPAPI [rtpHostBufferLock](#) (void *buffer, RTPsize size)
- [RTPresult](#) RTPAPI [rtpHostBufferUnlock](#) (void *buffer)
- [RTPresult](#) RTPAPI [rtpGetErrorString](#) ([RTPresult](#) errorCode, const char **errorString)
- [RTPresult](#) RTPAPI [rtpGetVersion](#) (unsigned *version)
- [RTPresult](#) RTPAPI [rtpGetVersionString](#) (const char **versionString)

3.9.2 Macro Definition Documentation

3.9.2.1 #define OPTIX_PRIME_VERSION

Value:

```
40000 /* major = OPTIX_PRIME_VERSION/10000, *
      * minor = (OPTIX_PRIME_VERSION%10000)/100, *
      * micro = OPTIX_PRIME_VERSION%100 */
```

3.9.3 Typedef Documentation

3.9.3.1 typedef struct RTPbufferdesc_api* RTPbufferdesc

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

3.9.3.2 typedef struct RTPcontext_api* RTPcontext

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

3.9.3.3 typedef struct RTPmodel_api* RTPmodel

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

3.9.3.4 typedef struct RTPquery_api* RTPquery

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

3.9.4 Function Documentation

3.9.4.1 RTPresult RTPAPI rtpBufferDescCreate (RTPcontext *context*, RTPbufferformat *format*, RTPbuffertype *type*, void * *buffer*, RTPbufferdesc * *desc*)

Create a buffer descriptor.

This function creates a buffer descriptor with the specified element format and buffertype. A buffer of type [RTP_BUFFER_TYPE_CUDA_LINEAR](#) is assumed to reside on the current device. The device number can be changed by calling [rtpBufferDescSetCudaDeviceNumber](#).

Parameters

| | | |
|-----|----------------|--------------------------------------|
| in | <i>context</i> | OptiX Prime context |
| in | <i>format</i> | Format of the buffer |
| in | <i>type</i> | Type of the buffer |
| in | <i>buffer</i> | Pointer to buffer data |
| out | <i>desc</i> | Pointer to the new buffer descriptor |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

Example Usage:

```
RTPbufferdesc verticesBD;
rtpBufferDescCreate(context, RTP_BUFFER_FORMAT_VERTEX_FLOAT3
, RTP_BUFFER_TYPE_HOST, vertices, &verticesBD);
```

3.9.4.2 RTPresult RTPAPI rtpBufferDescDestroy (RTPbufferdesc *desc*)

Destroys a buffer descriptor.

Buffer descriptors can be destroyed immediately after it is used as a function parameter. The buffer contents associated with a buffer descriptor, however, must remain valid until they are no longer used by any OptiX Prime objects.

Parameters

| | | |
|----|------|-------------------|
| in | desc | Buffer descriptor |
|----|------|-------------------|

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.3 RTPresult RTPAPI rtpBufferDescGetContext (RTPbufferdesc desc, RTPcontext * context)

Gets the context object associated with the provided buffer descriptor.

Parameters

| | | |
|-----|---------|-------------------|
| in | desc | Buffer descriptor |
| out | context | Returned context |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.4 RTPresult RTPAPI rtpBufferDescSetCudaDeviceNumber (RTPbufferdesc desc, unsigned deviceNumber)

Sets the CUDA device number for a buffer.

A buffer of type [RTP_BUFFER_TYPE_CUDA_LINEAR](#) is assumed to reside on the device that was current when its buffer descriptor was created unless otherwise specified using this function.

Parameters

| | | |
|----|--------------|--------------------|
| in | desc | Buffer descriptor |
| in | deviceNumber | CUDA device number |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.5 RTPresult RTPAPI rtpBufferDescSetRange (RTPbufferdesc desc, RTPsize begin, RTPsize end)

Sets the element range of a buffer to use.

The range is specified in terms of number of elements. By default, the range for a buffer is 0 to the number of elements in the buffer.

Parameters

| | | |
|----|--------------|--|
| in | <i>desc</i> | Buffer descriptor |
| in | <i>begin</i> | Start index of the range |
| in | <i>end</i> | End index of the range (exclusive, one past the index of the last element) |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.6 RTPresult RTPAPI rtpBufferDescSetStride (RTPbufferdesc *desc*, unsigned *strideBytes*)

Sets the stride for elements in a buffer.

This function is only valid for buffers of format [RTP_BUFFER_FORMAT_VERTEX_FLOAT3](#). This function is useful for vertex buffers that contain interleaved vertex attributes. For buffers that are transferred between the host and a device it is recommended that only buffers with default stride be used to avoid transferring data that will not be used.

Parameters

| | | |
|----|--------------------|---|
| in | <i>desc</i> | Buffer descriptor |
| in | <i>strideBytes</i> | Stride in bytes. The default value of 0 indicates that elements are contiguous in memory. |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

Example Usage:

```
struct Vertex {
    float3 pos, normal, color;
};
...
RTPbufferdesc vertsBD;
rtpBufferDescCreate(context, RTP_BUFFER_FORMAT_VERTEX_FLOAT3
    , RTP_BUFFER_TYPE_HOST, verts, &vertsBD);
rtpBufferDescSetRange(vertsBD, 0, numVerts);
rtpBufferDescSetStride(vertsBD, sizeof(Vertex));
```

3.9.4.7 RTPresult RTPAPI rtpContextCreate (RTPcontexttype *type*, RTPcontext * *context*)

Creates an OptiX Prime context.

By default, a context created with type [RTP_CONTEXT_TYPE_CUDA](#) will use all available CUDA devices. Specific devices can be selected using [rtpContextSetCudaDeviceNumbers](#). One device will be selected as the *primary device* and will be set as the current device when the function returns. If no available device has compute capability 2.0 or greater the created context will not be able to build acceleration structures.

Parameters

| | | |
|-----|----------------|--|
| in | <i>type</i> | The type of context to create |
| out | <i>context</i> | Pointer to the new OptiX Prime context |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_OBJECT_CREATION_FAILED](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_MEMORY_ALLOCATION_FAILED](#)

Example Usage:

```
RTPcontext context;
if(rtpContextCreate( RTP_CONTEXT_TYPE_CUDA, &context ) ==
    RTP_SUCCESS ) {
    int deviceNumbers[] = {0,1};
    rtpContextSetCudaDeviceNumbers( 2, deviceNumbers );
}
else
    rtpContextCreate( RTP_CONTEXT_TYPE_CPU, &context ); // Fallback to
    CPU
```

3.9.4.8 RTPresult RTPAPI rtpContextDestroy (RTPcontext context)

Destroys an OptiX Prime context.

Ongoing work is finished before *context* is destroyed. All OptiX Prime objects associated with *context* are also destroyed when *context* is destroyed.

Parameters

| | | |
|----|----------------|--------------------------------|
| in | <i>context</i> | OptiX Prime context to destroy |
|----|----------------|--------------------------------|

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.9 RTPresult RTPAPI rtpContextGetLastErrorString (RTPcontext context, const char ** return_string)

Returns a string describing last error encountered.

This function returns an error string for the last error encountered in *context* that may contain invocation-specific details beyond the simple [RTPresult](#) error code. Note that this function may return errors from previous asynchronous launches or from calls by other threads.

Parameters

| | | |
|-----|----------------------|---------------------------|
| in | <i>context</i> | OptiX Prime context |
| out | <i>return_string</i> | String with error details |

Return values

Relevant return values:

- [RTP_SUCCESS](#)

See also [rtpGetErrorString](#)

3.9.4.10 RTPresult RTPAPI rtpContextSetCpuThreads (RTPcontext *context*, unsigned *numThreads*)

Sets the number of CPU threads used by a CPU context.

This function will return an error if the provided *context* is not of type [RTP_CONTEXT_TYPE_CPU](#).

By default, one ray tracing thread is created per CPU core.

Parameters

| | | |
|----|-------------------|--|
| in | <i>context</i> | OptiX Prime context |
| in | <i>numThreads</i> | Number of threads used for the CPU context |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.11 RTPresult RTPAPI rtpContextSetCudaDeviceNumbers (RTPcontext *context*, unsigned *deviceCount*, const unsigned * *deviceNumbers*)

Sets the CUDA devices used by a context.

The first device provided in *deviceNumbers* will be used as the *primary device*. Acceleration structures will be built on the primary device and copied to the others. To build the acceleration structures the primary device must be of compute capability 2.0 or greater. The current device will be set to the primary device when this function returns.

If *deviceCount*==0, then the primary device is selected automatically and all available devices are selected for use. *deviceNumbers* is ignored.

Parameters

| | | |
|----|----------------------|--|
| in | <i>context</i> | OptiX Prime context |
| in | <i>deviceCount</i> | Number of devices supplied in <i>deviceNumbers</i> or 0 |
| in | <i>deviceNumbers</i> | Array of integer device indices, or NULL if <i>deviceCount</i> ==0 |

This function will return an error if the provided context is not of type [RTP_CONTEXT_TYPE_CUDA](#)

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.12 RTPresult RTPAPI rtpGetErrorString (RTPresult *errorCode*, const char ** *errorString*)

Translates an RTPresult error code to a string.

Translates an RTPresult error code to a string describing the error.

Parameters

| | | |
|-----|--------------------|-----------------------------|
| in | <i>errorCode</i> | Error code to be translated |
| out | <i>errorString</i> | Returned error string |

Return values

Relevant return values:

- [RTP_SUCCESS](#)

See also [rtpContextGetLastErrorString](#)

3.9.4.13 RTPResult RTPAPI rtpGetVersion (unsigned * *version*)

Gets OptiX Prime version number.

The encoding for the version number prior to OptiX 4.0.0 is $\text{major} \times 1000 + \text{minor} \times 10 + \text{micro}$. For versions 4.0.0 and higher, the encoding is $\text{major} \times 10000 + \text{minor} \times 100 + \text{micro}$. For example, for version 3.5.1 this function would return 3051, and for version 4.1.2 it would return 40102.

Parameters

| | | |
|-----|----------------|------------------|
| out | <i>version</i> | Returned version |
|-----|----------------|------------------|

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)

3.9.4.14 RTPResult RTPAPI rtpGetVersionString (const char ** *versionString*)

Gets OptiX Prime version string.

Returns OptiX Prime version string and other information in a human-readable format.

Parameters

| | | |
|----|----------------------|------------------------------|
| in | <i>versionString</i> | Returned version information |
|----|----------------------|------------------------------|

Return values

Relevant return values:

- [RTP_SUCCESS](#)

3.9.4.15 RTPResult RTPAPI rtpHostBufferLock (void * *buffer*, RTPsize *size*)

Page-locks a host buffer.

Transfers between the host and device are faster if the host buffers are page-locked. However, page-locked memory is a limited resource and should be used judiciously.

Parameters

| | | |
|----|---------------|--------------------|
| in | <i>buffer</i> | Buffer on the host |
| in | <i>size</i> | Size of the buffer |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)

3.9.4.16 RTPResult RTPAPI rtpHostBufferUnlock (void * *buffer*)

Unlocks a previously page-locked host buffer.

Transfers between the host and device are faster if the host buffers are page-locked. However, page-locked memory is a limited resource and should be used judiciously. Use this function on buffers previous page-locked with [rtpHostBufferLock](#).

Parameters

| | | |
|----|---------------|--------------------|
| in | <i>buffer</i> | Buffer on the host |
|----|---------------|--------------------|

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)

3.9.4.17 RTPResult RTPAPI rtpModelCopy (RTPmodel *model*, RTPmodel *srcModel*)

Copies one model to another.

This function copies a model from one OptiX Prime context to another for user-managed multi-GPU operation where one context is allocated per device. Only triangle models can be copied, not instance models. Furthermore, when a *srcModel* has the [RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES](#) build parameter set to 1, and it is intended that the triangle data is automatically transferred to the other context, the destination (*model*) should have the build parameter set to 0 before the copy call. If the destination model also has the build parameter set to 1, its triangles must be set by calling [rtpModelSetTriangles](#) followed by [rtpModelUpdate](#) using [RTP_MODEL_HINT_USER_TRIANGLES_AFTER_COPY_SET](#).

Parameters

| | | |
|----|-----------------|-------------------|
| in | <i>model</i> | Destination model |
| in | <i>srcModel</i> | Source model |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.18 RTPResult RTPAPI rtpModelCreate (RTPcontext *context*, RTPmodel * *model*)

Creates a model.

Parameters

| | | |
|-----|----------------|--------------------------|
| in | <i>context</i> | OptiX Prime context |
| out | <i>model</i> | Pointer to the new model |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.19 RTPResult RTPAPI rtpModelDestroy (RTPmodel *model*)

Destroys a model.

Any queries created on the model are also destroyed with the model. The queries are allowed to finish before they are destroyed.

Parameters

| | | |
|-----------|--------------|-------|
| <i>in</i> | <i>model</i> | Model |
|-----------|--------------|-------|

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.20 RTPresult RTPAPI rtpModelFinish (RTPmodel *model*)

Blocks current thread until model update is finished.

This function can be called multiple times. It will return immediately if the previous update has already finished.

Parameters

| | | |
|-----------|--------------|-------|
| <i>in</i> | <i>model</i> | Model |
|-----------|--------------|-------|

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.21 RTPresult RTPAPI rtpModelGetContext (RTPmodel *model*, RTPcontext * *context*)

Gets the context object associated with the model.

Parameters

| | | |
|------------|----------------|----------------------------------|
| <i>in</i> | <i>model</i> | Model to obtain the context from |
| <i>out</i> | <i>context</i> | Returned context |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.22 RTPresult RTPAPI rtpModelGetFinished (RTPmodel *model*, int * *isFinished*)

Polls the status of a model update.

Parameters

| | | |
|------------|-------------------|-------------------------|
| <i>in</i> | <i>model</i> | Model |
| <i>out</i> | <i>isFinished</i> | Returns finished status |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.23 RTPresult RTPAPI rtpModelSetBuilderParameter (RTPmodel *model_api*, RTPbuilderparam *param*, RTPsize *size*, const void * *ptr*)

Specifies a builder parameter for a model.

The following builder parameters are supported:

[RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES](#) : *int*

If the value for [RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES](#) is set to 0 (default), Prime uses an internal representation for triangles (which requires additional memory) to improve query performance and does not reference the user's vertex buffer during a query. If set to 1, Prime uses the provided triangle data as-is, which may result in slower query performance, but reduces memory usage.

[RTP_BUILDER_PARAM_CHUNK_SIZE](#) : *RTPsize*

Acceleration structures are built in chunks to reduce the amount of scratch memory needed. The size of the scratch memory chunk is specified in bytes by [RTP_BUILDER_PARAM_CHUNK_SIZE](#). If set to -1, the chunk size has no limit. If set to 0 (default) the chunk size is chosen automatically, currently as 10% of the total available video memory for GPU builds and 512MB for CPU builds.

Parameters

| | | |
|----|------------------|--|
| in | <i>model_api</i> | Model |
| in | <i>param</i> | Builder parameter to set |
| in | <i>size</i> | Size in bytes of the parameter being set |
| in | <i>ptr</i> | Pointer to where the value of the attribute will be copied from. This must point to at least <i>size</i> bytes of memory |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.24 RTPresult RTPAPI rtpModelSetInstances (RTPmodel *model*, RTPbufferdesc *instances*, RTPbufferdesc *transforms*)

Sets the instance data for a model.

The *instances* buffer specifies a list of model instances, and the *transforms* buffer holds a transformation matrix for each instance. The instance buffer type must be [RTP_BUFFER_TYPE_HOST](#).

Instance buffers must be of format [RTP_BUFFER_FORMAT_INSTANCE_MODEL](#), and transform buffers of format [RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x4](#) or [RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x3](#). If a stride is specified for the transformations, it must be a multiple of 16 bytes. Furthermore, the matrices must be stored in row-major order. Only affine transformations are supported, and the last row is always assumed to be [0.0, 0.0, 0.0, 1.0].

All instance models in the *instances* buffer must belong to the same context as the model itself. Additionally, the build parameter [RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES](#) must be the same for all models (if applied). Setting [RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES](#) for a model which contains instances has no effect.

The buffers are not used until [rtpModelUpdate](#) is called.

Parameters

| | | |
|----|--------------|-------|
| in | <i>model</i> | Model |
|----|--------------|-------|

| | | |
|----|-------------------|--|
| in | <i>instances</i> | Buffer descriptor for instances |
| in | <i>transforms</i> | Buffer descriptor for 4x4 transform matrices |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.25 RTPResult RTPAPI rtpModelSetTriangles (RTPmodel *model*, RTPbufferdesc *indices*, RTPbufferdesc *vertices*)

Sets the triangle data for a model.

The index buffer specifies triplet of vertex indices. If the index buffer descriptor is not specified (e.g. *indices*==NULL), the vertex buffer is considered to be a flat list of triangles, with every three vertices forming a triangle. The buffers are not used until [rtpModelUpdate](#) is called.

Parameters

| | | |
|----|-----------------|--|
| in | <i>model</i> | Model |
| in | <i>indices</i> | Buffer descriptor for triangle vertex indices, or NULL |
| in | <i>vertices</i> | Buffer descriptor for triangle vertices |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.26 RTPResult RTPAPI rtpModelUpdate (RTPmodel *model*, unsigned *hints*)

Updates data, or creates an acceleration structure over triangles or instances.

Depending on the specified hints, [rtpModelUpdate](#) performs different operations:

If the flag [RTP_MODEL_HINT_ASYNC](#) is specified, some or all of the acceleration structure update may run asynchronously and [rtpModelUpdate](#) may return before the update is finished. In the case of [RTP_MODEL_HINT_NONE](#), the acceleration structure build is blocking. It is important that buffers specified in [rtpModelSetTriangles](#) and [rtpModelSetInstances](#) not be modified until the update has finished. [rtpModelFinish](#) blocks the current thread until the update is finished. [rtpModelGetFinished](#) can be used to poll until the update is finished. Once the update has finished the input buffers can be modified.

The acceleration structure build performed by [rtpModelUpdate](#) uses a fast, high quality algorithm, but has the cost of requiring additional working memory. The amount of working memory is controlled by [RTP_BUILDER_PARAM_CHUNK_SIZE](#).

The flag [RTP_MODEL_HINT_MASK_UPDATE](#) should be used to inform Prime when visibility mask data changed (after calling [rtpModelSetTriangles](#) with the updated values), e.g. when the indices format [RTP_BUFFER_FORMAT_INDICES_INT3_MASK_INT](#) is used. [RTP_MODEL_HINT_MASK_UPDATE](#) can be combined with [RTP_MODEL_HINT_ASYNC](#) to perform asynchronous data updates.

Hint [RTP_MODEL_HINT_USER_TRIANGLES_AFTER_COPY_SET](#) should be used when a triangle model has been copied (with the user triangle build flag set), and new user triangles have been set (by calling [rtpModelSetTriangles](#) again with the updated values). [RTP_MODEL_HINT_USER_TRIANGLES_AFTER_COPY_SET](#) can be combined with [RTP_MODEL_HINT_ASYNC](#) to perform asynchronous data updates.

Parameters

| | | |
|----|--------------|--|
| in | <i>model</i> | Model |
| in | <i>hints</i> | A combination of flags from RTPmodelhint |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

Example Usage:

```
RTPmodel model;
rtpModelCreate(context, &model);
rtpModelSetTriangles(model, 0, vertsBD);
rtpModelUpdate(model, RTP_MODEL_HINT_ASYNC);

// ... do useful work on CPU while GPU is busy

rtpModelFinish(model);

// It is now safe to modify vertex buffer
```

3.9.4.27 RTPresult RTPAPI rtpQueryCreate (RTPmodel *model*, RTPquerytype *queryType*, RTPquery * *query*)

Creates a query on a model.

If the model to which a query is bound destroyed with [rtpModelDestroy\(\)](#) the query will be destroyed as well.

Parameters

| | | |
|-----|------------------|-----------------------------|
| in | <i>model</i> | Model to use for this query |
| in | <i>queryType</i> | Type of the query |
| out | <i>query</i> | Pointer to the new query |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.28 RTPresult RTPAPI rtpQueryDestroy (RTPquery *query*)

Destroys a query.

The query is finished before it is destroyed

Parameters

| | | |
|----|--------------|-----------------------|
| in | <i>query</i> | Query to be destroyed |
|----|--------------|-----------------------|

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.29 RTPResult RTPAPI rtpQueryExecute (RTPQuery *query*, unsigned *hints*)

Executes a raytracing query.

If the flag [RTP_QUERY_HINT_ASYNC](#) is specified, `rtpQueryExecute` may return before the query is actually finished. [rtpQueryFinish](#) can be called to block the current thread until the query is finished, or [rtpQueryGetFinished](#) can be used to poll until the query is finished.

Parameters

| | | |
|----|--------------|--|
| in | <i>query</i> | Query |
| in | <i>hints</i> | A combination of flags from RTPQueryhint |

Once the query has finished all of the hits are guaranteed to have been returned, and it is safe to modify the ray buffer.

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

Example Usage:

```
RTPQuery query;
rtpQueryCreate(model, RTP_QUERY_TYPE_CLOSEST, &query);
rtpQuerySetRays(query, raysBD);
rtpQuerySetHits(hits, hitsBD);
rtpQueryExecute(query, 0);
// safe to modify ray buffer and process hits
```

3.9.4.30 RTPResult RTPAPI rtpQueryFinish (RTPQuery *query*)

Blocks current thread until query is finished.

This function can be called multiple times. It will return immediately if the query has already finished.

Parameters

| | | |
|----|--------------|-------|
| in | <i>query</i> | Query |
|----|--------------|-------|

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.31 RTPResult RTPAPI rtpQueryGetContext (RTPQuery *query*, RTPcontext * *context*)

Gets the context object associated with a query.

Parameters

| | | |
|-----|----------------|----------------------------------|
| in | <i>query</i> | Query to obtain the context from |
| out | <i>context</i> | Returned context |

Return values

Relevant return values:

- [RTP_SUCCESS](#)

- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.32 RTPResult RTPAPI rtpQueryGetFinished (RTPQuery query, int * isFinished)

Polls the status of a query.

Parameters

| | | |
|-----|-------------------|-------------------------|
| in | <i>query</i> | Query |
| out | <i>isFinished</i> | Returns finished status |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.33 RTPResult RTPAPI rtpQuerySetCudaStream (RTPQuery query, cudaStream_t stream)

Sets a sync stream for a query.

Specify a Cuda stream used for synchronization. If no stream is specified, the default 0-stream is used. A stream can only be specified for contexts with type [RTP_CONTEXT_TYPE_CUDA](#).

Parameters

| | | |
|----|---------------|---------------|
| in | <i>query</i> | Query |
| in | <i>stream</i> | A cuda stream |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.34 RTPResult RTPAPI rtpQuerySetHits (RTPQuery query, RTPbufferdesc hits)

Sets the hits buffer for a query.

A hit is reported for every ray in the query. Therefore the size of the range in the hit buffer must match that of the ray buffer.

Parameters

| | | |
|----|--------------|----------------------------|
| in | <i>query</i> | Query |
| in | <i>hits</i> | Buffer descriptor for hits |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.9.4.35 RTPresult RTPAPI rtpQuerySetRays (RTPQuery *query*, RTPbufferdesc *rays*)

Sets the rays buffer for a query.

The rays buffer is not accessed until [rtpQueryExecute\(\)](#) is called. The ray directions must be unit length for correct results.

Parameters

| | | |
|----|--------------|----------------------------|
| in | <i>query</i> | Query |
| in | <i>rays</i> | Buffer descriptor for rays |

Return values

Relevant return values:

- [RTP_SUCCESS](#)
- [RTP_ERROR_INVALID_VALUE](#)
- [RTP_ERROR_UNKNOWN](#)

3.10 optix_prime_declarations.h File Reference

3.10.1 Detailed Description

OptiX Prime public API declarations.

Author

NVIDIA Corporation OptiX Prime public API declarations

Enumerations

- enum RTPresult {
[RTP_SUCCESS](#) = 0,
[RTP_ERROR_INVALID_VALUE](#) = 1,
[RTP_ERROR_OUT_OF_MEMORY](#) = 2,
[RTP_ERROR_INVALID_HANDLE](#) = 3,
[RTP_ERROR_NOT_SUPPORTED](#) = 4,
[RTP_ERROR_OBJECT_CREATION_FAILED](#) = 5,
[RTP_ERROR_MEMORY_ALLOCATION_FAILED](#) = 6,
[RTP_ERROR_INVALID_CONTEXT](#) = 7,
[RTP_ERROR_VALIDATION_ERROR](#) = 8,
[RTP_ERROR_INVALID_OPERATION](#) = 9,
[RTP_ERROR_UNKNOWN](#) = 999 }
- enum RTPcontexttype {
[RTP_CONTEXT_TYPE_CPU](#) = 0x100,
[RTP_CONTEXT_TYPE_CUDA](#) = 0x101 }
- enum RTPbuffertype {
[RTP_BUFFER_TYPE_HOST](#) = 0x200,
[RTP_BUFFER_TYPE_CUDA_LINEAR](#) = 0x201 }
- enum RTPbufferformat {


```

RTP_BUFFER_FORMAT_INDICES_INT3 = 0x400,
RTP_BUFFER_FORMAT_INDICES_INT3_MASK_INT = 0x401,
RTP_BUFFER_FORMAT_VERTEX_FLOAT3 = 0x420,
RTP_BUFFER_FORMAT_VERTEX_FLOAT4 = 0x421,
RTP_BUFFER_FORMAT_RAY_ORIGIN_DIRECTION = 0x440,
RTP_BUFFER_FORMAT_RAY_ORIGIN_TMIN_DIRECTION_TMAX = 0x441,
RTP_BUFFER_FORMAT_RAY_ORIGIN_MASK_DIRECTION_TMAX = 0x442,
RTP_BUFFER_FORMAT_HIT_BITMASK = 0x460,
RTP_BUFFER_FORMAT_HIT_T = 0x461,
RTP_BUFFER_FORMAT_HIT_T_TRIID = 0x462,
RTP_BUFFER_FORMAT_HIT_T_TRIID_U_V = 0x463,
RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID = 0x464,
RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID_U_V = 0x465,
RTP_BUFFER_FORMAT_INSTANCE_MODEL = 0x480,
RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x4 = 0x490,
RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x3 = 0x491 }

• enum RTPQuerytype {
    RTP_QUERY_TYPE_ANY = 0x1000,
    RTP_QUERY_TYPE_CLOSEST = 0x1001 }

• enum RTPmodelhint {
    RTP_MODEL_HINT_NONE = 0x0000,
    RTP_MODEL_HINT_ASYNC = 0x2001,
    RTP_MODEL_HINT_MASK_UPDATE = 0x2002,
    RTP_MODEL_HINT_USER_TRIANGLES_AFTER_COPY_SET = 0x2004 }

• enum RTPQueryhint {
    RTP_QUERY_HINT_NONE = 0x0000,
    RTP_QUERY_HINT_ASYNC = 0x4001,
    RTP_QUERY_HINT_WATERTIGHT = 0x4002 }

• enum RTPbuilderparam {
    RTP_BUILDER_PARAM_CHUNK_SIZE = 0x800,
    RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES = 0x801 }

```

3.10.2 Enumeration Type Documentation

3.10.2.1 enum RTPbufferformat

Buffer formats

Enumerator

RTP_BUFFER_FORMAT_INDICES_INT3 Index buffer with 3 integer vertex indices per triangle

RTP_BUFFER_FORMAT_INDICES_INT3_MASK_INT Index buffer with 3 integer vertex indices per triangle, and an integer visibility mask

RTP_BUFFER_FORMAT_VERTEX_FLOAT3 Vertex buffer with 3 floats per vertex position

RTP_BUFFER_FORMAT_VERTEX_FLOAT4 Vertex buffer with 4 floats per vertex position

RTP_BUFFER_FORMAT_RAY_ORIGIN_DIRECTION float3:origin float3:direction

RTP_BUFFER_FORMAT_RAY_ORIGIN_TMIN_DIRECTION_TMAX float3:origin, float:tmin, float3:direction, float:tmax

RTP_BUFFER_FORMAT_RAY_ORIGIN_MASK_DIRECTION_TMAX float3:origin, int:mask, float3:direction, float:tmax. If used, buffer format RTP_BUFFER_FORMAT_INDICES_INT3_MASK_INT is required!

RTP_BUFFER_FORMAT_HIT_BITMASK one bit per ray 0=miss, 1=hit

RTP_BUFFER_FORMAT_HIT_T float:ray distance (t < 0 for miss)

RTP_BUFFER_FORMAT_HIT_T_TRIID float:ray distance (t < 0 for miss), int:triangle id

RTP_BUFFER_FORMAT_HIT_T_TRIID_U_V float:ray distance (t < 0 for miss), int:triangle id, float2:barycentric coordinates u,v (w=1-u-v)

`RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID` float:ray distance ($t < 0$ for miss), int:triangle id, int:instance position in list

`RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID_U_V` float:ray distance ($t < 0$ for miss), int:triangle id, int:instance position in list, float2:barycentric coordinates u,v ($w=1-u-v$)

`RTP_BUFFER_FORMAT_INSTANCE_MODEL` RTPmodel:objects of type RTPmodel

`RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x4` float:row major 4x4 affine matrix (it is assumed that the last row has the entries 0.0f, 0.0f, 0.0f, 1.0f, and will be ignored)

`RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x3` float:row major 4x3 affine matrix

3.10.2.2 enum RTPbuffertype

Buffer types

Enumerator

`RTP_BUFFER_TYPE_HOST` Buffer in host memory

`RTP_BUFFER_TYPE_CUDA_LINEAR` Linear buffer in device memory on a cuda device

3.10.2.3 enum RTPbuilderparam

Enumerator

`RTP_BUILDER_PARAM_CHUNK_SIZE` Number of bytes used for a chunk of the acceleration structure build

`RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES` A hint to specify which data should be used for the intersection test

3.10.2.4 enum RTPcontexttype

Context types

Enumerator

`RTP_CONTEXT_TYPE_CPU` CPU context

`RTP_CONTEXT_TYPE_CUDA` CUDA context

3.10.2.5 enum RTPmodelhint

Model hints

Enumerator

`RTP_MODEL_HINT_NONE` No hints. Use default settings.

`RTP_MODEL_HINT_ASYNC` Asynchronous model updating

`RTP_MODEL_HINT_MASK_UPDATE` Upload buffer with mask data again

`RTP_MODEL_HINT_USER_TRIANGLES_AFTER_COPY_SET` Clear dirty flag of triangles.

3.10.2.6 enum RTPqueryhint

Query hints

Enumerator

`RTP_QUERY_HINT_NONE` No hints. Use default settings.

`RTP_QUERY_HINT_ASYNC` Asynchronous query execution

`RTP_QUERY_HINT_WATERTIGHT` Use watertight ray-triangle intersection, but only if the `RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES` builder parameter is also set

3.10.2.7 enum `RTPQuerytype`

Query types

Enumerator

`RTP_QUERY_TYPE_ANY` Return any hit along a ray

`RTP_QUERY_TYPE_CLOSEST` Return only the closest hit along a ray

3.10.2.8 enum `RTPResult`

Return value for OptiX Prime APIs

Enumerator

`RTP_SUCCESS` Success

`RTP_ERROR_INVALID_VALUE` An invalid value was provided

`RTP_ERROR_OUT_OF_MEMORY` Out of memory

`RTP_ERROR_INVALID_HANDLE` An invalid handle was supplied

`RTP_ERROR_NOT_SUPPORTED` An unsupported function was requested

`RTP_ERROR_OBJECT_CREATION_FAILED` Object creation failed

`RTP_ERROR_MEMORY_ALLOCATION_FAILED` Memory allocation failed

`RTP_ERROR_INVALID_CONTEXT` An invalid context was provided

`RTP_ERROR_VALIDATION_ERROR` A validation error occurred

`RTP_ERROR_INVALID_OPERATION` An invalid operation was performed

`RTP_ERROR_UNKNOWN` Unknown error

3.11 `optix_primepp.h` File Reference

3.11.1 Detailed Description

A C++ wrapper around the OptiX Prime API.

Classes

- class `optix::prime::ContextObj`
- class `optix::prime::BufferDescObj`
- class `optix::prime::ModelObj`
- class `optix::prime::QueryObj`
- class `optix::prime::Exception`

Macros

- `#define CHK(code) checkError(code, getContext()->getRTPcontext())`

Typedefs

- typedef `Handle< BufferDescObj > optix::prime::BufferDesc`
- typedef `Handle< ContextObj > optix::prime::Context`
- typedef `Handle< ModelObj > optix::prime::Model`
- typedef `Handle< QueryObj > optix::prime::Query`

Functions

- `std::string optix::prime::getVersionString ()`
- `void optix::prime::checkError (RTPresult code)`
- `void optix::prime::checkError (RTPresult code, RTPcontext context)`

3.12 `optix_world.h` File Reference

3.12.1 Detailed Description

OptiX public API C and C++ API.

Author

NVIDIA Corporation This header is designed to be included by both host and device code providing access to the C-API along with the C++ API found in `optixpp_namespaces.h`. In addition various helper classes and file will also be included when compiling C++ compatible code.

Note that the CUDA vector types will be defined in the `optix::` namespace.

3.13 `optixpp_namespace.h` File Reference

3.13.1 Detailed Description

A C++ wrapper around the OptiX API.

Classes

- class `optix::Handle< T >`
- class `optix::Exception`
- class `optix::APIObj`
- class `optix::DestroyableObj`
- class `optix::ScopedObj`
- class `optix::VariableObj`
- class `optix::ContextObj`
- class `optix::ProgramObj`
- class `optix::GroupObj`
- class `optix::GeometryGroupObj`
- class `optix::TransformObj`
- class `optix::SelectorObj`
- class `optix::AccelerationObj`
- class `optix::GeometryInstanceObj`
- class `optix::GeometryObj`
- class `optix::MaterialObj`
- class `optix::TextureSamplerObj`
- class `optix::BufferObj`
- struct `optix::bufferId< T, Dim >`
- class `optix::callableProgramId< T >`
- class `optix::RemoteDeviceObj`

Macros

- `#define WIN32_LEAN_AND_MEAN`
- `#define rtBufferId optix::bufferId`
- `#define RT_INTERNAL_CALLABLE_PROGRAM_DEFS()`
- `#define rtCallableProgramId optix::callableProgramId`

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< RemoteDeviceObj > **optix::RemoteDevice**
- typedef Handle< SelectorObj > **optix::Selector**
- typedef Handle< TextureSamplerObj > **optix::TextureSampler**
- typedef Handle< TransformObj > **optix::Transform**
- typedef Handle< VariableObj > **optix::Variable**

Functions

- template<typename ReturnT >
 class callableProgramId< ReturnT()> **optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()**
- template<typename ReturnT , typename Arg0T >
 class callableProgramId
 < ReturnT(Arg0T)> **optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T >
 class callableProgramId
 < ReturnT(Arg0T, Arg1T)> **optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T >
 class callableProgramId
 < ReturnT(Arg0T, Arg1T, Arg2T)> **optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T >
 class callableProgramId
 < ReturnT(Arg0T, Arg1T, Arg2T, Arg3T)> **optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T >
 class callableProgramId
 < ReturnT(Arg0T, Arg1T, Arg2T, Arg3T, Arg4T)> **optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T >
 class callableProgramId
 < ReturnT(Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T)> **optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T >
 class callableProgramId
 < ReturnT(Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T)> **optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()**
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T , typename Arg7T >
 class callableProgramId
 < ReturnT(Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T)> **optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()**

- `template<typename ReturnT, typename Arg0T, typename Arg1T, typename Arg2T, typename Arg3T, typename Arg4T, typename Arg5T, typename Arg6T, typename Arg7T, typename Arg8T >`
`class callableProgramId`
`< ReturnT(Arg0T, Arg1T, Arg2T,`
`Arg3T, Arg4T, Arg5T, Arg6T,`
`Arg7T, Arg8T)> optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()`
- `template<typename ReturnT, typename Arg0T, typename Arg1T, typename Arg2T, typename Arg3T, typename Arg4T, typename Arg5T, typename Arg6T, typename Arg7T, typename Arg8T, typename Arg9T >`
`class callableProgramId`
`< ReturnT(Arg0T, Arg1T, Arg2T,`
`Arg3T, Arg4T, Arg5T, Arg6T,`
`Arg7T, Arg8T, Arg9T)> optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()`

3.13.2 Macro Definition Documentation

3.13.2.1 #define RT_INTERNAL_CALLABLE_PROGRAM_DEFS()

Value:

```
{
    public:
        callableProgramId() {}
        callableProgramId(int id) : m_id(id) {}
        int getId() const { return m_id; }
    private:
        int m_id;
}
```

`callableProgramId` is a host version of the device side `callableProgramId`.

Use `callableProgramId` to define types that can be included from both the host and device code. This class provides a container that can be used to transport the program id back and forth between host and device code. The `callableProgramId` class is useful, because it can take a program id obtained from `rtProgramGetId` and provide accessors for calling the program corresponding to the program id.

"`bindless_type.h`" used by both host and device code:

```
#include <optix_world.h>
struct ProgramInfo {
    int val;
    rtProgramId<int(int)> program;
};
```

Host code:

```
#include "bindless_type.h"
ProgramInfo input_program_info;
input_program_info.val = 0;
input_program_info.program = rtCallableProgramId<int(int)>(inputProgram0->getId());
context["input_program_info"]->setUserData(sizeof(ProgramInfo), &input_program_info);
```

Device code:

```
#include "bindless_type.h"
rtBuffer<int,1> result;
rtDeclareVariable(ProgramInfo, input_program_info, ,);

RT_PROGRAM void bindless()
{
    int value = input_program_info.program(input_program_info.val);
    result[0] = value;
}
```

3.14 optixu.h File Reference

3.14.1 Detailed Description

Simple API for performing raytracing queries using OptiX or the CPU.

Macros

- `#define RTU_INLINE` static 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` (`RObjectType` 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 RTAPI rtuCreateClusteredMesh` (`RTcontext` context, `unsigned int usePTX32InHost64`, `RTgeometry *mesh`, `unsigned int num_verts`, `const float *verts`, `unsigned int num_tris`, `const unsigned *indices`, `const unsigned *mat_indices`)
- `RTresult RTAPI rtuCreateClusteredMeshExt` (`RTcontext` context, `unsigned int usePTX32InHost64`, `RTgeometry *mesh`, `unsigned int num_verts`, `const float *verts`, `unsigned int num_tris`, `const unsigned *indices`, `const unsigned *mat_indices`, `RTbuffer` norms, `const unsigned *norm_indices`, `RTbuffer` tex_coords, `const unsigned *tex_indices`)
- `RTU_INLINE RTresult rtuGroupAddChild` (`RTgroup` group, `RObject` child, `unsigned int *index`)
- `RTU_INLINE RTresult rtuSelectorAddChild` (`RTselector` selector, `RObject` child, `unsigned int *index`)
- `RTU_INLINE RTresult rtuGeometryGroupAddChild` (`RTgeometrygroup` geometrygroup, `RTgeometryinstance` child, `unsigned int *index`)
- `RTU_INLINE RTresult rtuTransformSetChild` (`RTtransform` transform, `RObject` child)
- `RTU_INLINE RTresult rtuTransformGetChild` (`RTtransform` transform, `RObject *type`)
- `RTU_INLINE RTresult rtuTransformGetChildType` (`RTtransform` transform, `RObjectType *type`)
- `RTU_INLINE RTresult rtuGroupRemoveChild` (`RTgroup` group, `RObject` child)
- `RTU_INLINE RTresult rtuSelectorRemoveChild` (`RTselector` selector, `RObject` child)
- `RTU_INLINE 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, `RObject` child, `unsigned int *index`)
- `RTU_INLINE RTresult rtuSelectorGetChildIndex` (`RTselector` selector, `RObject` child, `unsigned int *index`)
- `RTU_INLINE RTresult rtuGeometryGroupGetChildIndex` (`RTgeometrygroup` geometrygroup, `RTgeometryinstance` child, `unsigned int *index`)

3.14.2 Macro Definition Documentation

3.14.2.1 #define RTU_CHECK_ERROR(func)

Value:

```
do {
    RTresult code = func;
    if( code != RT_SUCCESS )
        return code;
} while(0)
```

3.14.2.2 #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
```

3.14.2.3 #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
```

3.14.3 Function Documentation

3.14.3.1 RTresult RTAPI rtuCreateClusteredMesh (RTcontext context, unsigned int usePTX32InHost64, RTgeometry * mesh, unsigned int num_verts, const float * verts, unsigned int num_tris, const unsigned * indices, const unsigned * mat_indices)

Create clustered triangle mesh for good memory coherence with paging on. Vertex, index and material buffers are created and attached to the mesh. Cluster's bounding box and intersection programs are attached to the mesh. The intersection program has the following attributes:

- `rtDeclareVariable(int, primitive_id, attribute primitive_id,);`
- `rtDeclareVariable(float3, texcoord, attribute texcoord,);` It is always zero
- `rtDeclareVariable(float3, geometric_normal, attribute geometric_normal,);`
- `rtDeclareVariable(float3, shading_normal, attribute shading_normal,);` It is equal to geometric_normal

Created `RTgeometry` mesh expects there to be placed into a `RTgeometryinstance` where the `mat_indices` specified map into materials attached to the `RTgeometryinstance`

In the event of an error, please query the error string from the `RTcontext`.

Parameters

| | |
|--------------------------|---|
| <i>context</i> | Context |
| <i>usePTX32In-Host64</i> | Use 32bit PTX bounding box and intersection programs in 64bit application. Takes effect only with 64bit host. |
| <i>mesh</i> | Output geometry |
| <i>num_verts</i> | Vertex count |
| <i>verts</i> | Vertices (num_verts*float*3) [v1_x, v1_y, v1_z, v2.x, ...] |
| <i>num_tris</i> | Triangle count |
| <i>indices</i> | Vertex indices (num_tris*unsigned*3) [tri1_index1, tri1_index2, ...] |
| <i>mat_indices</i> | Indices of materials (num_tris*unsigned) [tri1_mat_index, tri2_mat_index, ...] |

3.14.3.2 RTresult RTAPI rtuCreateClusteredMeshExt (RTcontext context, unsigned int usePTX32InHost64, RTgeometry * mesh, unsigned int num_verts, const float * verts, unsigned int num_tris, const unsigned * indices, const unsigned * mat_indices, RTbuffer norms, const unsigned * norm_indices, RTbuffer tex_coords, const unsigned * tex_indices)

Create clustered triangle mesh for good memory coherence with paging on. Buffers for vertices, indices, normals, indices of normals, texture coordinates, indices of texture coordinates and materials are created and attached to the mesh. Cluster's bounding box and intersection programs are attached to the mesh. The intersection program has the following attributes:

- [rtDeclareVariable\(int, primitive_id, attribute primitive_id, \);](#)
- [rtDeclareVariable\(float3, texcoord, attribute texcoord, \);](#)
- [rtDeclareVariable\(float3, geometric_normal, attribute geometric_normal, \);](#)
- [rtDeclareVariable\(float3, shading_normal, attribute shading_normal, \);](#)

Created [RTgeometry](#) mesh expects there to be placed into a [RTgeometryinstance](#) where the mat_indices specified map into materials attached to the [RTgeometryinstance](#)

Vertex, normal and texture coordinate buffers can be shared between many geometry objects

In the event of an error, please query the error string from the RTcontext.

Parameters

| | |
|--------------------------|---|
| <i>context</i> | Context |
| <i>usePTX32In-Host64</i> | Use 32bit PTX bounding box and intersection programs in 64bit application. Takes effect only with 64bit host. |
| <i>mesh</i> | Output geometry |
| <i>num_verts</i> | Vertex count |
| <i>verts</i> | Vertices (num_verts*float*3) [v1_x, v1_y, v1_z, v2.x, ...] |
| <i>num_tris</i> | Triangle count |
| <i>indices</i> | Vertex indices (num_tris*unsigned*3) [tri1_index1, tri1_index2, ...] |
| <i>mat_indices</i> | Indices of materials (num_tris*unsigned) [tri1_mat_index, tri2_mat_index, ...] |
| <i>norms</i> | Normals (num_norms*float*3) [v1_x, v1_y, v1_z, v2.x, ...] |
| <i>norm_indices</i> | Indices of vertex normals (num_tris*unsigned*3) [tri1_norm_index1, tri1_norm_index2 ...] |
| <i>tex_coords</i> | Texture uv coords (num_tex_coords*float*2) [t1_u, t1_v, t2_u ...] |
| <i>tex_indices</i> | Indices of texture uv (num_tris*unsigned*3) [tri1_tex_index1, tri1_tex_index2 ...] |

3.14.3.3 RTresult RTAPI rtuCUDACompileFile (const char * filename, const char ** preprocessorArguments, unsigned int numPreprocessorArguments, RTsize * resultSize, RTsize * errorSize)

Compile a cuda source file.

Parameters

| | | |
|-----|-----------------------------------|--|
| in | <i>filename</i> | source code file name |
| in | <i>preprocessor-Arguments</i> | list of preprocessor arguments |
| in | <i>num-Preprocessor-Arguments</i> | number of preprocessor arguments |
| out | <i>resultSize</i> | size required to hold compiled result string |
| out | <i>errorSize</i> | size required to hold error string |

Return values

| | |
|-----------------|-------------|
| <i>RTresult</i> | Return code |
|-----------------|-------------|

3.14.3.4 RTresult RTAPI rtuCUDACompileString (const char * *source*, const char ** *preprocessorArguments*, unsigned int *numPreprocessorArguments*, RTsize * *resultSize*, RTsize * *errorSize*)

Compile a cuda source string.

Parameters

| | | |
|-----|-----------------------------------|--|
| in | <i>source</i> | source code string |
| in | <i>preprocessor-Arguments</i> | list of preprocessor arguments |
| in | <i>num-Preprocessor-Arguments</i> | number of preprocessor arguments |
| out | <i>resultSize</i> | size required to hold compiled result string |
| out | <i>errorSize</i> | size required to hold error string |

Return values

| | |
|-----------------|-------------|
| <i>RTresult</i> | Return code |
|-----------------|-------------|

3.14.3.5 RTresult RTAPI rtuCUDAGetCompileResult (char * *result*, char * *error*)

Get the result of the most recent call to one of the above compile functions. The 'result' and 'error' parameters must point to memory large enough to hold the respective strings, as returned by the compile function.

Parameters

| | | |
|-----|---------------|------------------------|
| out | <i>result</i> | compiled result string |
| out | <i>error</i> | error string |

Return values

| | |
|-----------------|-------------|
| <i>RTresult</i> | Return code |
|-----------------|-------------|

3.14.3.6 RTU_INLINE RTresult rtuGeometryGroupAddChild (RTgeometrygroup *geometrygroup*, RTgeometryinstance *child*, unsigned int * *index*)

Add an entry to the end of the child array. Fills 'index' with the index of the added child, if the pointer is non-NULL.

3.14.3.7 RTU_INLINE RTresult rtuGeometryGroupGetChildIndex (RTgeometrygroup *geometrygroup*, RTgeometryinstance *child*, unsigned int * *index*)

Use a linear search to find the child in the child array, and return its index. Returns [RT_SUCCESS](#) if the child was found, [RT_ERROR_INVALID_VALUE](#) otherwise.

3.14.3.8 RTU_INLINE RTresult rtuGeometryGroupRemoveChild (RTgeometrygroup *geometrygroup*, RTgeometryinstance *child*)

Find the given child using a linear search in the child array and remove it. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

3.14.3.9 RTU_INLINE RTresult rtuGeometryGroupRemoveChildByIndex (RTgeometrygroup *geometrygroup*, unsigned int *index*)

Remove the child at the given index in the child array. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

3.14.3.10 RTresult RTAPI rtuGetSizeForRTformat (RTformat *format*, size_t * *size*)

Return the size of a given [RTformat](#). [RT_FORMAT_USER](#) and [RT_FORMAT_UNKNOWN](#) return 0. Returns [RT_ERROR_INVALID_VALUE](#) if the format isn't recognized, [RT_SUCCESS](#) otherwise.

Parameters

| | | |
|-----|---------------|--------------------|
| in | <i>format</i> | OptiX format |
| out | <i>size</i> | Size of the format |

Return values

| | |
|-----------------|-------------|
| <i>RTresult</i> | Return code |
|-----------------|-------------|

3.14.3.11 RTU_INLINE RTresult rtuGroupAddChild (RTgroup *group*, RObject *child*, unsigned int * *index*)

Add an entry to the end of the child array. Fills 'index' with the index of the added child, if the pointer is non-NULL.

3.14.3.12 RTU_INLINE RTresult rtuGroupGetChildIndex (RTgroup *group*, RObject *child*, unsigned int * *index*)

Use a linear search to find the child in the child array, and return its index. Returns [RT_SUCCESS](#) if the child was found, [RT_ERROR_INVALID_VALUE](#) otherwise.

3.14.3.13 RTU_INLINE RTresult rtuGroupRemoveChild (RTgroup *group*, RObject *child*)

Find the given child using a linear search in the child array and remove it. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

3.14.3.14 RTU_INLINE RTresult rtuGroupRemoveChildByIndex (RTgroup *group*, unsigned int *index*)

Remove the child at the given index in the child array. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

3.14.3.15 RTresult RTAPI rtuNameForType (RObjecttype *type*, char * *buffer*, RTsize *bufferSize*)

Get the name string of a given type. See [RObjecttype](#) for more information.

Parameters

| | | |
|-----|-------------------|----------------------------------|
| in | <i>type</i> | Type requested |
| out | <i>buffer</i> | Buffer to output the name string |
| in | <i>bufferSize</i> | Size of the provided buffer |

Return values

| | |
|-----------------|-------------|
| <i>RTresult</i> | Return code |
|-----------------|-------------|

3.14.3.16 RTU_INLINE RTresult rtuSelectorAddChild (RTselector *selector*, RObject *child*, unsigned int * *index*)

Add an entry to the end of the child array. Fills 'index' with the index of the added child, if the pointer is non-NULL.

3.14.3.17 `RTU_INLINE RTresult rtuSelectorGetChildIndex (RTselector selector, RObject child, unsigned int * index)`

Use a linear search to find the child in the child array, and return its index. Returns `RT_SUCCESS` if the child was found, `RT_ERROR_INVALID_VALUE` otherwise.

3.14.3.18 `RTU_INLINE RTresult rtuSelectorRemoveChild (RTselector selector, RObject child)`

Find the given child using a linear search in the child array and remove it. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

3.14.3.19 `RTU_INLINE RTresult rtuSelectorRemoveChildByIndex (RTselector selector, unsigned int index)`

Remove the child at the given index in the child array. If it's not the last entry in the child array, the last entry in the array will replace the deleted entry, in order to shrink the array size by one.

3.14.3.20 `RTU_INLINE RTresult rtuTransformGetChild (RTtransform transform, RObject * type)`

Wrap `rtTransformGetChild` and `rtTransformGetChildType` in order to provide a type-safe version for C++.

3.14.3.21 `RTU_INLINE RTresult rtuTransformGetChildType (RTtransform transform, RObjecttype * type)`

Wrap `rtTransformGetChild` and `rtTransformGetChildType` in order to provide a type-safe version for C++.

3.14.3.22 `RTU_INLINE RTresult rtuTransformSetChild (RTtransform transform, RObject child)`

Wrap `rtTransformSetChild` in order to provide a type-safe version for C++.

3.15 optixu_aabb_namespace.h File Reference

3.15.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Public AABB namespace

Classes

- class [optix::Aabb](#)

Macros

- `#define RT_AABB_ASSERT assert`
- `#define OPTIXU_INLINE_DEFINED 1`
- `#define OPTIXU_INLINE __forceinline__`

3.16 optixu_math_namespace.h File Reference

3.16.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation This file implements common mathematical operations on vector types (float3, float4 etc.) since these are not provided as standard by CUDA.

The syntax is modelled on the Cg standard library.

This file has also been modified from the original cutil_math.h file. cutil_math.h is a subset of this file, and you should use this file in place of any cutil_math.h file you wish to use.

Classes

- struct [optix::Onb](#)

Macros

- #define **OPTIXU_INLINE_DEFINED** 1
- #define **OPTIXU_INLINE** __forceinline__
- #define **OPTIXU_MATH_DEFINE_IN_NAMESPACE**

Typedefs

- typedef unsigned int **optix::uint**
- typedef unsigned short **optix::ushort**

Functions

- OPTIXU_INLINE float **optix::fminf** (const float a, const float b)
- OPTIXU_INLINE float **optix::fmaxf** (const float a, const float b)
- OPTIXU_INLINE float **optix::copysignf** (const float dst, const float src)
- OPTIXU_INLINE int **optix::max** (int a, int b)
- OPTIXU_INLINE int **optix::min** (int a, int b)
- OPTIXU_INLINE int **optix::float_as_int** (const float f)
- OPTIXU_INLINE float **optix::int_as_float** (int i)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::lerp** (const float a, const float b, const float t)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::bilerp** (const float x00, const float x10, const float x01, const float x11, const float u, const float v)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::clamp** (const float f, const float a, const float b)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::getByIndex** (const float1 &v, int i)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (float1 &v, int i, float x)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator-** (const float2 &a)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::lerp** (const float2 &a, const float2 &b, const float t)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::bilerp** (const float2 &x00, const float2 &x10, const float2 &x01, const float2 &x11, const float u, const float v)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::dot** (const float2 &a, const float2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::length** (const float2 &v)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::normalize** (const float2 &v)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::floor** (const float2 &v)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::reflect** (const float2 &i, const float2 &n)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::faceforward** (const float2 &n, const float2 &i, const float2 &nref)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::expf** (const float2 &v)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::getByIndex** (const float2 &v, int i)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (float2 &v, int i, float x)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator-** (const float3 &a)

- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::lerp** (const float3 &a, const float3 &b, const float t)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::bilerp** (const float3 &x00, const float3 &x10, const float3 &x01, const float3 &x11, const float u, const float v)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::dot** (const float3 &a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::cross** (const float3 &a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::length** (const float3 &v)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::normalize** (const float3 &v)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::floor** (const float3 &v)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::reflect** (const float3 &i, const float3 &n)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::faceforward** (const float3 &n, const float3 &i, const float3 &nref)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::expf** (const float3 &v)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::getByIndex** (const float3 &v, int i)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (float3 &v, int i, float x)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator-** (const float4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::lerp** (const float4 &a, const float4 &b, const float t)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::bilerp** (const float4 &x00, const float4 &x10, const float4 &x01, const float4 &x11, const float u, const float v)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::dot** (const float4 &a, const float4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::length** (const float4 &r)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::normalize** (const float4 &v)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::floor** (const float4 &v)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::reflect** (const float4 &i, const float4 &n)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::faceforward** (const float4 &n, const float4 &i, const float4 &nref)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::expf** (const float4 &v)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::getByIndex** (const float4 &v, int i)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (float4 &v, int i, float x)
- OPTIXU_INLINE RT_HOSTDEVICE int **optix::clamp** (const int f, const int a, const int b)
- OPTIXU_INLINE RT_HOSTDEVICE int **optix::getByIndex** (const int1 &v, int i)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (int1 &v, int i, int x)
- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::operator-** (const int2 &a)
- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::min** (const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::max** (const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int **optix::getByIndex** (const int2 &v, int i)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (int2 &v, int i, int x)
- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::operator-** (const int3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::min** (const int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::max** (const int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int **optix::getByIndex** (const int3 &v, int i)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (int3 &v, int i, int x)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::operator-** (const int4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::min** (const int4 &a, const int4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::max** (const int4 &a, const int4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int **optix::getByIndex** (const int4 &v, int i)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (int4 &v, int i, int x)
- OPTIXU_INLINE RT_HOSTDEVICE
unsigned int **optix::clamp** (const unsigned int f, const unsigned int a, const unsigned int b)
- OPTIXU_INLINE RT_HOSTDEVICE
unsigned int **optix::getByIndex** (const uint1 &v, unsigned int i)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (uint1 &v, int i, unsigned int x)
- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::min** (const uint2 &a, const uint2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::max** (const uint2 &a, const uint2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE
unsigned int **optix::getByIndex** (const uint2 &v, unsigned int i)

- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (uint2 &v, int i, unsigned int x)
- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::min** (const uint3 &a, const uint3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::max** (const uint3 &a, const uint3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE
unsigned int **optix::getByIndex** (const uint3 &v, unsigned int i)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (uint3 &v, int i, unsigned int x)
- OPTIXU_INLINE RT_HOSTDEVICE
unsigned int **optix::getByIndex** (const uint4 &v, unsigned int i)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::setByIndex** (uint4 &v, int i, unsigned int x)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::smoothstep** (const float edge0, const float edge1, const float x)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::temperature** (const float t)
- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::intersect_triangle_branchless** (const Ray &ray, const float3 &p0, const float3 &p1, const float3 &p2, float3 &n, float &t, float &beta, float &gamma)
- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::intersect_triangle_earlyexit** (const Ray &ray, const float3 &p0, const float3 &p1, const float3 &p2, float3 &n, float &t, float &beta, float &gamma)
- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::intersect_triangle** (const Ray &ray, const float3 &p0, const float3 &p1, const float3 &p2, float3 &n, float &t, float &beta, float &gamma)
- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::refract** (float3 &r, const float3 &i, const float3 &n, const float ior)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::fresnel_schlick** (const float cos_theta, const float exponent=5.0f, const float minimum=0.0f, const float maximum=1.0f)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::fresnel_schlick** (const float cos_theta, const float exponent, const float3 &minimum, const float3 &maximum)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::luminance** (const float3 &rgb)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::luminanceCIE** (const float3 &rgb)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::cosine_sample_hemisphere** (const float u1, const float u2, float3 &p)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::square_to_disk** (const float2 &sample)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::cart_to_pol** (const float3 &v)

- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::make_float2** (const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::make_float2** (const int2 &a)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::make_float2** (const uint2 &a)

- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::fminf** (const float2 &a, const float2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::fminf** (const float2 &a)

- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::fmaxf** (const float2 &a, const float2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::fmaxf** (const float2 &a)

- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator+** (const float2 &a, const float2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator+** (const float2 &a, const float b)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator+** (const float a, const float2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator+=** (float2 &a, const float2 &b)

- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator-** (const float2 &a, const float2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator-** (const float2 &a, const float b)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator-** (const float a, const float2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator-=** (float2 &a, const float2 &b)

- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator*** (const float2 &a, const float2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator*** (const float2 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator*** (const float s, const float2 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (float2 &a, const float2 &s)

- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (float2 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator/** (const float2 &a, const float2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator/** (const float2 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::operator/** (const float s, const float2 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator/=** (float2 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::clamp** (const float2 &v, const float a, const float b)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::clamp** (const float2 &v, const float2 &a, const float2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::make_float3** (const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::make_float3** (const float2 &a)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::make_float3** (const int3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::make_float3** (const uint3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::fminf** (const float3 &a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::fminf** (const float3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::fmaxf** (const float3 &a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::fmaxf** (const float3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator+** (const float3 &a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator+** (const float3 &a, const float b)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator+** (const float a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator+=** (float3 &a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator-** (const float3 &a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator-** (const float3 &a, const float b)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator-** (const float a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator-=** (float3 &a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator*** (const float3 &a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator*** (const float3 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator*** (const float s, const float3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (float3 &a, const float3 &s)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (float3 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator/** (const float3 &a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator/** (const float3 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator/** (const float s, const float3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator/=** (float3 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::clamp** (const float3 &v, const float a, const float b)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::clamp** (const float3 &v, const float3 &a, const float3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::make_float4** (const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::make_float4** (const float3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::make_float4** (const int4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::make_float4** (const uint4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::fminf** (const float4 &a, const float4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::fminf** (const float4 &a)

- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::fmaxf** (const float4 &a, const float4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float **optix::fmaxf** (const float4 &a)

- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator+** (const float4 &a, const float4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator+** (const float4 &a, const float b)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator+** (const float a, const float4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator+=** (float4 &a, const float4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator-** (const float4 &a, const float4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator-** (const float4 &a, const float b)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator-** (const float a, const float4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator-=** (float4 &a, const float4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator*** (const float4 &a, const float4 &s)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator*** (const float4 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator*** (const float s, const float4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (float4 &a, const float4 &s)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (float4 &a, const float s)

- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator/** (const float4 &a, const float4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator/** (const float4 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator/** (const float s, const float4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator/=** (float4 &a, const float s)

- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::clamp** (const float4 &v, const float a, const float b)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::clamp** (const float4 &v, const float4 &a, const float4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::make_int2** (const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::make_int2** (const float2 &a)

- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::operator+** (const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator+=** (int2 &a, const int2 &b)

- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::operator-** (const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::operator-** (const int2 &a, const int b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator-=** (int2 &a, const int2 &b)

- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::operator*** (const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::operator*** (const int2 &a, const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::operator*** (const int s, const int2 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (int2 &a, const int s)

- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::clamp** (const int2 &v, const int a, const int b)
- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::clamp** (const int2 &v, const int2 &a, const int2 &b)

- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator==** (const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator!=** (const int2 &a, const int2 &b)

- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::make_int3** (const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::make_int3** (const float3 &a)

- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::operator+** (const int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator+=** (int3 &a, const int3 &b)

- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::operator-** (const int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator-=** (int3 &a, const int3 &b)

- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::operator*** (const int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::operator*** (const int3 &a, const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::operator*** (const int s, const int3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (int3 &a, const int s)

- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::operator/** (const int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::operator/** (const int3 &a, const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::operator/** (const int s, const int3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator/=** (int3 &a, const int s)

- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::clamp** (const int3 &v, const int a, const int b)
- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::clamp** (const int3 &v, const int3 &a, const int3 &b)

- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator==** (const int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator!=** (const int3 &a, const int3 &b)

- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::make_int4** (const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::make_int4** (const float4 &a)

- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::operator+** (const int4 &a, const int4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator+=** (int4 &a, const int4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::operator-** (const int4 &a, const int4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator-=** (int4 &a, const int4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::operator*** (const int4 &a, const int4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::operator*** (const int4 &a, const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::operator*** (const int s, const int4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (int4 &a, const int s)

- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::operator/** (const int4 &a, const int4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::operator/** (const int4 &a, const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::operator/** (const int s, const int4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator/=** (int4 &a, const int s)

- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::clamp** (const int4 &v, const int a, const int b)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::clamp** (const int4 &v, const int4 &a, const int4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator==** (const int4 &a, const int4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator!=** (const int4 &a, const int4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::make_uint2** (const unsigned int s)
- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::make_uint2** (const float2 &a)

- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::operator+** (const uint2 &a, const uint2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator+=** (uint2 &a, const uint2 &b)

- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::operator-** (const uint2 &a, const uint2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::operator-** (const uint2 &a, const unsigned int b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator-=** (uint2 &a, const uint2 &b)

- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::operator*** (const uint2 &a, const uint2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::operator*** (const uint2 &a, const unsigned int s)
- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::operator*** (const unsigned int s, const uint2 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (uint2 &a, const unsigned int s)

- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::clamp** (const uint2 &v, const unsigned int a, const unsigned int b)
- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::clamp** (const uint2 &v, const uint2 &a, const uint2 &b)

- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator==** (const uint2 &a, const uint2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator!=** (const uint2 &a, const uint2 &b)

- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::make_uint3** (const unsigned int s)
- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::make_uint3** (const float3 &a)

- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::operator+** (const uint3 &a, const uint3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator+=** (uint3 &a, const uint3 &b)

- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::operator-** (const uint3 &a, const uint3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator-=** (uint3 &a, const uint3 &b)

- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::operator*** (const uint3 &a, const uint3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::operator*** (const uint3 &a, const unsigned int s)
- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::operator*** (const unsigned int s, const uint3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (uint3 &a, const unsigned int s)

- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::operator/** (const uint3 &a, const uint3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::operator/** (const uint3 &a, const unsigned int s)
- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::operator/** (const unsigned int s, const uint3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator/=** (uint3 &a, const unsigned int s)

- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::clamp** (const uint3 &v, const unsigned int a, const unsigned int b)
- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::clamp** (const uint3 &v, const uint3 &a, const uint3 &b)

- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator==** (const uint3 &a, const uint3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator!=** (const uint3 &a, const uint3 &b)

- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::make_uint4** (const unsigned int s)
- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::make_uint4** (const float4 &a)

- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::min** (const uint4 &a, const uint4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::max** (const uint4 &a, const uint4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::operator+** (const uint4 &a, const uint4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator+=** (uint4 &a, const uint4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::operator-** (const uint4 &a, const uint4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator-=** (uint4 &a, const uint4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::operator*** (const uint4 &a, const uint4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::operator*** (const uint4 &a, const unsigned int s)

- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::operator*** (const unsigned int s, const uint4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator*=** (uint4 &a, const unsigned int s)

- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::operator/** (const uint4 &a, const uint4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::operator/** (const uint4 &a, const unsigned int s)
- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::operator/** (const unsigned int s, const uint4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void **optix::operator/=** (uint4 &a, const unsigned int s)

- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::clamp** (const uint4 &v, const unsigned int a, const unsigned int b)
- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::clamp** (const uint4 &v, const uint4 &a, const uint4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator==** (const uint4 &a, const uint4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE bool **optix::operator!=** (const uint4 &a, const uint4 &b)

- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::make_int2** (const int3 &v0)
- OPTIXU_INLINE RT_HOSTDEVICE int2 **optix::make_int2** (const int4 &v0)
- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::make_int3** (const int4 &v0)
- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::make_uint2** (const uint3 &v0)
- OPTIXU_INLINE RT_HOSTDEVICE uint2 **optix::make_uint2** (const uint4 &v0)
- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::make_uint3** (const uint4 &v0)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::make_float2** (const float3 &v0)
- OPTIXU_INLINE RT_HOSTDEVICE float2 **optix::make_float2** (const float4 &v0)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::make_float3** (const float4 &v0)

- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::make_int3** (const int v0, const int2 &v1)
- OPTIXU_INLINE RT_HOSTDEVICE int3 **optix::make_int3** (const int2 &v0, const int v1)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::make_int4** (const int v0, const int v1, const int2 &v2)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::make_int4** (const int v0, const int2 &v1, const int v2)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::make_int4** (const int2 &v0, const int v1, const int v2)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::make_int4** (const int v0, const int3 &v1)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::make_int4** (const int3 &v0, const int v1)
- OPTIXU_INLINE RT_HOSTDEVICE int4 **optix::make_int4** (const int2 &v0, const int2 &v1)
- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::make_uint3** (const unsigned int v0, const uint2 &v1)
- OPTIXU_INLINE RT_HOSTDEVICE uint3 **optix::make_uint3** (const uint2 &v0, const unsigned int v1)
- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::make_uint4** (const unsigned int v0, const unsigned int v1, const uint2 &v2)
- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::make_uint4** (const unsigned int v0, const uint2 &v1, const unsigned int v2)
- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::make_uint4** (const uint2 &v0, const unsigned int v1, const unsigned int v2)
- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::make_uint4** (const unsigned int v0, const uint3 &v1)
- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::make_uint4** (const uint3 &v0, const unsigned int v1)
- OPTIXU_INLINE RT_HOSTDEVICE uint4 **optix::make_uint4** (const uint2 &v0, const uint2 &v1)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::make_float3** (const float2 &v0, const float v1)
- OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::make_float3** (const float v0, const float2 &v1)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::make_float4** (const float v0, const float v1, const float2 &v2)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::make_float4** (const float v0, const float2 &v1, const float v2)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::make_float4** (const float2 &v0, const float v1, const float v2)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::make_float4** (const float v0, const float3 &v1)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::make_float4** (const float3 &v0, const float v1)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::make_float4** (const float2 &v0, const float2 &v1)

3.17 optixu_math_stream_namespace.h File Reference

3.17.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation Stream operators for CUDA vector types

Functions

- `std::ostream & optix::operator<< (std::ostream &os, const optix::Aabb &aabb)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::float4 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::float4 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::float3 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::float3 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::float2 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::float2 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::int4 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::int4 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::int3 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::int3 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::int2 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::int2 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::uint4 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::uint4 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::uint3 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::uint3 &v)`
- `std::ostream & optix::operator<< (std::ostream &os, const optix::uint2 &v)`
- `std::istream & optix::operator>> (std::istream &is, optix::uint2 &v)`
- `template<unsigned int M, unsigned int N>
std::ostream & optix::operator<< (std::ostream &os, const optix::Matrix< M, N > &m)`
- `template<unsigned int M, unsigned int N>
std::istream & optix::operator>> (std::istream &is, optix::Matrix< M, N > &m)`

3.18 optixu_matrix_namespace.h File Reference

3.18.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Public Matrix namespace

Classes

- struct [optix::VectorDim](#)< DIM >
- struct [optix::VectorDim](#)< 2 >
- struct [optix::VectorDim](#)< 3 >
- struct [optix::VectorDim](#)< 4 >
- class [optix::Matrix](#)< M, N >
- class [optix::Matrix](#)< M, N >

Macros

- `#define OPTIXU_INLINE_DEFINED 1`
- `#define OPTIXU_INLINE __forceinline__`
- `#define RT_MATRIX_ACCESS(m, i, j) m[i*N+j]`
- `#define RT_MAT_DECL template <unsigned int M, unsigned int N>`

Typedefs

- `typedef Matrix< 2, 2 > optix::Matrix2x2`
- `typedef Matrix< 2, 3 > optix::Matrix2x3`
- `typedef Matrix< 2, 4 > optix::Matrix2x4`
- `typedef Matrix< 3, 2 > optix::Matrix3x2`
- `typedef Matrix< 3, 3 > optix::Matrix3x3`
- `typedef Matrix< 3, 4 > optix::Matrix3x4`
- `typedef Matrix< 4, 2 > optix::Matrix4x2`
- `typedef Matrix< 4, 3 > optix::Matrix4x3`
- `typedef Matrix< 4, 4 > optix::Matrix4x4`

Functions

- `template<unsigned int M>`
`OPTIXU_INLINE RT_HOSTDEVICE`
`Matrix< M, M > & optix::operator*= (Matrix< M, M > &m1, const Matrix< M, M > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE`
`RT_HOSTDEVICE Matrix< M, N > & optix::operator= (Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE`
`RT_HOSTDEVICE Matrix< M, N > & optix::operator+= (Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE`
`RT_HOSTDEVICE Matrix< M, N > & optix::operator*= (Matrix< M, N > &m1, float f)`
- `RT_MAT_DECL OPTIXU_INLINE`
`RT_HOSTDEVICE Matrix< M, N > & optix::operator/= (Matrix< M, N > &m1, float f)`
- `RT_MAT_DECL OPTIXU_INLINE`
`RT_HOSTDEVICE Matrix< M, N > optix::operator- (const Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE`
`RT_HOSTDEVICE Matrix< M, N > optix::operator+ (const Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE`
`RT_HOSTDEVICE Matrix< M, N > optix::operator/ (const Matrix< M, N > &m, float f)`
- `RT_MAT_DECL OPTIXU_INLINE`
`RT_HOSTDEVICE Matrix< M, N > optix::operator* (const Matrix< M, N > &m, float f)`
- `RT_MAT_DECL OPTIXU_INLINE`
`RT_HOSTDEVICE Matrix< M, N > optix::operator* (float f, const Matrix< M, N > &m)`
- `RT_MAT_DECL OPTIXU_INLINE`
`RT_HOSTDEVICE Matrix< M, N >`
`::floatM optix::operator* (const Matrix< M, N > &m, const typename Matrix< M, N >::floatN &v)`
- `RT_MAT_DECL OPTIXU_INLINE`
`RT_HOSTDEVICE Matrix< M, N >`
`::floatN optix::operator* (const typename Matrix< M, N >::floatM &v, const Matrix< M, N > &m)`
- `template<unsigned int M, unsigned int N, unsigned int R>`
`OPTIXU_INLINE RT_HOSTDEVICE`
`Matrix< M, R > optix::operator* (const Matrix< M, N > &m1, const Matrix< N, R > &m2)`
- `template<unsigned int N>`
`OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator* (const Matrix< 2, N > &m, const typename`
`Matrix< 2, N >::floatN &vec)`

- template<unsigned int N>
OPTIXU_INLINE RT_HOSTDEVICE float3 **optix::operator*** (const Matrix< 3, N > &m, const typename Matrix< 3, N >::floatN &vec)
- template<unsigned int N>
OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator*** (const Matrix< 4, N > &m, const typename Matrix< 4, N >::floatN &vec)
- OPTIXU_INLINE RT_HOSTDEVICE float4 **optix::operator*** (const Matrix< 4, 4 > &m, const float4 &vec)
- template<unsigned int M, unsigned int N, unsigned int R>
RT_HOSTDEVICE Matrix< M, R > **optix::operator*** (const Matrix< M, N > &m1, const Matrix< N, R > &m2)
- template<unsigned int M>
RT_HOSTDEVICE Matrix< M, M > & **optix::operator*=** (Matrix< M, M > &m1, const Matrix< M, M > &m2)
- OPTIXU_INLINE RT_HOSTDEVICE
Matrix< 3, 3 > **optix::make_matrix3x3** (const Matrix< 4, 4 > &matrix)

3.19 optixu_traversal.h File Reference

3.19.1 Detailed Description

Simple API for performing raytracing queries using OptiX or the CPU.

Classes

- struct [RTUtraversalresult](#)

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)

3.19.2 Typedef Documentation

3.19.2.1 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.

3.19.3 Enumeration Type Documentation

3.19.3.1 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](#)** No option
- [RTU_INITOPTION_GPU_ONLY](#)** GPU only
- [RTU_INITOPTION_CPU_ONLY](#)** CPU only
- [RTU_INITOPTION_CULL_BACKFACE](#)** Back face culling

3.19.3.2 enum [RTUoption](#)

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

Enumerator

- [RTU_OPTION_INT_NUM_THREADS](#)** Number of threads

3.19.3.3 enum RTUoutput

RTUoutput requested.

Enumerator

RTU_OUTPUT_NONE Output None
RTU_OUTPUT_NORMAL float3 [x, y, z]
RTU_OUTPUT_BARYCENTRIC float2 [alpha, beta] (gamma implicit)
RTU_OUTPUT_BACKFACING char [1 | 0]

3.19.3.4 enum RTUquerytype

The type of ray query to be performed.

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

Enumerator

RTU_QUERY_TYPE_ANY_HIT Perform any hit calculation
RTU_QUERY_TYPE_CLOSEST_HIT Perform closest hit calculation
RTU_QUERY_TYPE_COUNT Query type count

3.19.3.5 enum RTUrayformat

The input format of the ray vector.

Enumerator

RTU_RAYFORMAT_ORIGIN_DIRECTION_TMIN_TMAX_INTERLEAVED Origin Direction Tmin Tmax interleaved
RTU_RAYFORMAT_ORIGIN_DIRECTION_INTERLEAVED Origin Direction interleaved
RTU_RAYFORMAT_COUNT [Ray](#) format count

3.19.3.6 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 Triangle format mesh
RTU_TRIFORMAT_TRIANGLE_SOUP Triangle 'soup' format
RTU_TRIFORMAT_COUNT Triangle format count

3.19.4 Function Documentation

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

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

Parameters

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

3.19.4.2 RTResult RTAPI rtuTraversalDestroy (RTUtraversal *traversal*)

Clean up any internal memory associated with *rtuTraversal** operations. Includes destruction of result buffers returned via [rtuTraversalGetErrorString](#). Invalidates traversal object.

Parameters

| | |
|------------------|------------------------|
| <i>traversal</i> | Traversal state handle |
|------------------|------------------------|

3.19.4.3 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

| | | |
|-----|------------------|------------------------|
| | <i>traversal</i> | Traversal state handle |
| out | <i>data</i> | Acceleration data |

3.19.4.4 RTResult RTAPI rtuTraversalGetAccelDataSize (RTUtraversal *traversal*, RTsize * *data_size*)

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

Parameters

| | | |
|-----|------------------|---------------------------|
| | <i>traversal</i> | Traversal state handle |
| out | <i>data_size</i> | Size of acceleration data |

3.19.4.5 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.

For a list of associated error codes that this function might inspect take a look at [RTresult](#) .

Parameters

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

3.19.4.6 RTResult RTAPI rtuTraversalMapOutput (RTUtraversal *traversal*, RTUoutput *which*, void ** *output*)

Retrieve user-specified output from last [rtuTraversalTraverse](#) 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 [rtuTraversalMapRays](#). 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 [rtuTraversalCreate](#) an error will be returned. See [RTUoutput](#) enum for description of output data formats for various outputs.

Parameters

| | | |
|-----|------------------|--------------------------------------|
| | <i>traversal</i> | Traversal state handle |
| | <i>which</i> | Output type to be specified |
| out | <i>output</i> | Pointer to output from last traverse |

3.19.4.7 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

| | | |
|--|------------------|-----------------------------|
| | <i>traversal</i> | Traversal state handle |
| | <i>num_rays</i> | Number of rays to be traced |
| | <i>rays</i> | Pointer to ray data |

3.19.4.8 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

| | | |
|-----|------------------|-------------------------------------|
| | <i>traversal</i> | Traversal state handle |
| out | <i>results</i> | Pointer to results of last traverse |

3.19.4.9 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

| | | |
|--|------------------|------------------------|
| | <i>traversal</i> | Traversal state handle |
|--|------------------|------------------------|

3.19.4.10 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

| | | |
|--|------------------|---------------------------|
| | <i>traversal</i> | Traversal state handle |
| | <i>data</i> | Acceleration data |
| | <i>data_size</i> | Size of acceleration data |

3.19.4.11 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 [rtuTraversalTraverse](#). 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

| | |
|------------------|---|
| <i>traversal</i> | Traversal state handle |
| <i>num_verts</i> | Vertex count |
| <i>verts</i> | Vertices [v1_x, v1_y, v1_z, v2.x, ...] |
| <i>num_tris</i> | Triangle count |
| <i>indices</i> | Indices [tri1_index1, tri1_index2, ...] |

3.19.4.12 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

| | |
|------------------|------------------------|
| <i>traversal</i> | Traversal state handle |
| <i>option</i> | The option to be set |
| <i>value</i> | Value of the option |

3.19.4.13 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

| | |
|------------------|---|
| <i>traversal</i> | Traversal state handle |
| <i>num_tris</i> | Triangle count |
| <i>tris</i> | Triangles [tri1_v1.x, tri1_v1.y, tri1_v1.z, tri1_v2.x, ...] |

3.19.4.14 RTresult RTAPI rtuTraversalTraverse (RTUtraversal *traversal*)

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

Parameters

| | |
|------------------|------------------------|
| <i>traversal</i> | Traversal state handle |
|------------------|------------------------|

3.19.4.15 RTresult RTAPI rtuTraversalUnmapOutput (RTUtraversal *traversal*, RTUoutput *which*)

See [rtuTraversalMapOutput](#) .

3.19.4.16 RTresult RTAPI rtuTraversalUnmapRays (RTUtraversal *traversal*)

See [rtuTraversalMapRays](#) .

3.19.4.17 RTresult RTAPI rtuTraversalUnmapResults (RTUtraversal *traversal*)

See [rtuTraversalMapResults](#) .

Index

- ~Exception
 - optix::Exception, [22](#)
- Aabb
 - optix::Aabb, [4](#)
- area
 - optix::Aabb, [4](#)
- bindProgressiveStream
 - optix::BufferObj, [12](#)
- center
 - optix::Aabb, [4](#)
- checkError
 - optix::APIObj, [8](#)
 - optix::ContextObj, [17](#)
- contains
 - optix::Aabb, [4](#)
- create1DLayeredBuffer
 - optix::ContextObj, [17](#)
- create2DLayeredBuffer
 - optix::ContextObj, [17](#)
- createAcceleration
 - optix::ContextObj, [17](#)
- createBuffer
 - optix::ContextObj, [17](#), [18](#)
- createBufferForCUDA
 - optix::ContextObj, [18](#)
- createCubeBuffer
 - optix::ContextObj, [18](#)
- createCubeLayeredBuffer
 - optix::ContextObj, [18](#)
- createGeometryGroup
 - optix::ContextObj, [18](#)
- createGeometryInstance
 - optix::ContextObj, [18](#)
- createGroup
 - optix::ContextObj, [18](#)
- createMipmappedBuffer
 - optix::ContextObj, [18](#), [19](#)
- DXGI Texture Formats, [2](#)
- declareVariable
 - optix::ContextObj, [19](#)
 - optix::GeometryInstanceObj, [25](#)
 - optix::GeometryObj, [26](#)
 - optix::MaterialObj, [30](#)
 - optix::ProgramObj, [36](#)
 - optix::ScopedObj, [42](#)
- det
 - optix::Matrix, [32](#)
- direction
 - Ray, [39](#)
- distance
 - optix::Aabb, [4](#)
- distance2
 - optix::Aabb, [4](#)
- enlarge
 - optix::Aabb, [4](#)
- extent
 - optix::Aabb, [4](#)
- fromBasis
 - optix::Matrix, [32](#)
- getBufferFromId
 - optix::ContextObj, [19](#)
- getCol
 - optix::Matrix, [32](#)
- getContext
 - optix::ContextObj, [19](#)
- getData
 - optix::Matrix, [32](#), [33](#)
- getDataSize
 - optix::AccelerationObj, [7](#)
- getDevicePointer
 - optix::BufferObj, [12](#)
- getEnabledDeviceCount
 - optix::ContextObj, [19](#)
- getGLBOld
 - optix::BufferObj, [12](#)
- getId
 - optix::BufferObj, [12](#)
 - optix::ProgramObj, [37](#)
 - optix::TextureSamplerObj, [45](#)
- getMaxTextureCount
 - optix::ContextObj, [19](#)
- getPrimitiveCount
 - optix::GeometryObj, [26](#)
- getPrimitiveIndexOffset
 - optix::GeometryObj, [26](#)
- getProgramFromId
 - optix::ContextObj, [19](#)
- getProperty
 - optix::AccelerationObj, [7](#)
- getRow
 - optix::Matrix, [33](#)
- getTextureSamplerFromId
 - optix::ContextObj, [19](#)
- getVariableCount
 - optix::ContextObj, [19](#)
 - optix::GeometryInstanceObj, [25](#)
 - optix::GeometryObj, [26](#)
 - optix::MaterialObj, [30](#)
 - optix::ProgramObj, [37](#)
 - optix::ScopedObj, [42](#)
- halfArea
 - optix::Aabb, [4](#)
- identity

- optix::Matrix, [33](#)
- include
 - optix::Aabb, [4](#), [5](#)
- intersection
 - optix::Aabb, [5](#)
- intersects
 - optix::Aabb, [5](#)
- invalidate
 - optix::Aabb, [5](#)
- inverse
 - optix::Matrix, [33](#)
- isFlat
 - optix::Aabb, [5](#)
- launch
 - optix::ContextObj, [19](#)
- launchProgressive
 - optix::ContextObj, [20](#)
- longestAxis
 - optix::Aabb, [5](#)
- m_max
 - optix::Aabb, [6](#)
- m_min
 - optix::Aabb, [6](#)
- makeException
 - optix::Exception, [22](#)
- map
 - optix::BufferObj, [12](#)
- markDirty
 - optix::AccelerationObj, [7](#)
 - optix::GeometryObj, [26](#)
- Matrix
 - optix::Matrix, [32](#)
- maxExtent
 - optix::Aabb, [5](#)
- OPTIX_VERSION
 - optix.h, [52](#)
- OpenGL Texture Formats, [1](#)
- operator<
 - optix::Matrix, [33](#)
- operator=
 - optix::Matrix, [33](#)
- operator==
 - optix::Aabb, [5](#)
- optix.h, [52](#)
 - OPTIX_VERSION, [52](#)
- optix_declarations.h
 - RT_BUFFER_ATTRIBUTE_STREAM_BITRATE, [60](#)
 - RT_BUFFER_ATTRIBUTE_STREAM_FORMAT, [60](#)
 - RT_BUFFER_ATTRIBUTE_STREAM_FPS, [60](#)
 - RT_BUFFER_ATTRIBUTE_STREAM_GAMMA, [60](#)
 - RT_BUFFER_COPY_ON_DIRTY, [60](#)
 - RT_BUFFER_CUBEMAP, [60](#)
 - RT_BUFFER_GPU_LOCAL, [60](#)
 - RT_BUFFER_ID_NULL, [60](#)
 - RT_BUFFER_INPUT, [61](#)
 - RT_BUFFER_INPUT_OUTPUT, [61](#)
 - RT_BUFFER_LAYERED, [60](#)
 - RT_BUFFER_MAP_READ, [61](#)
 - RT_BUFFER_MAP_READ_WRITE, [61](#)
 - RT_BUFFER_MAP_WRITE, [61](#)
 - RT_BUFFER_MAP_WRITE_DISCARD, [61](#)
 - RT_BUFFER_OUTPUT, [61](#)
 - RT_BUFFER_PROGRESSIVE_STREAM, [61](#)
 - RT_CONTEXT_ATTRIBUTE_AVAILABLE_DEVICE_MEMORY, [61](#)
 - RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS, [61](#)
 - RT_CONTEXT_ATTRIBUTE_GPU_PAGING_ACTIVE, [61](#)
 - RT_CONTEXT_ATTRIBUTE_GPU_PAGING_FORCED_OFF, [61](#)
 - RT_CONTEXT_ATTRIBUTE_MAX_TEXTURE_COUNT, [61](#)
 - RT_CONTEXT_ATTRIBUTE_USED_HOST_MEMORY, [61](#)
 - RT_DEVICE_ATTRIBUTE_CLOCK_RATE, [61](#)
 - RT_DEVICE_ATTRIBUTE_COMPUTE_CAPABILITY, [61](#)
 - RT_DEVICE_ATTRIBUTE_CUDA_DEVICE_ORDINAL, [61](#)
 - RT_DEVICE_ATTRIBUTE_EXECUTION_TIMEOUT_ENABLED, [61](#)
 - RT_DEVICE_ATTRIBUTE_MAX_HARDWARE_TEXTURE_COUNT, [61](#)
 - RT_DEVICE_ATTRIBUTE_MAX_THREADS_PER_BLOCK, [61](#)
 - RT_DEVICE_ATTRIBUTE_MULTIPROCESSOR_COUNT, [61](#)
 - RT_DEVICE_ATTRIBUTE_NAME, [61](#)
 - RT_DEVICE_ATTRIBUTE_TCC_DRIVER, [61](#)
 - RT_DEVICE_ATTRIBUTE_TOTAL_MEMORY, [61](#)
 - RT_ERROR_ALREADY_MAPPED, [65](#)
 - RT_ERROR_AUTHENTICATION_FAILED, [65](#)
 - RT_ERROR_CLUSTER_ALREADY_RUNNING, [65](#)
 - RT_ERROR_CLUSTER_NOT_RUNNING, [65](#)
 - RT_ERROR_CONNECTION_ALREADY_EXISTS, [65](#)
 - RT_ERROR_CONNECTION_FAILED, [65](#)
 - RT_ERROR_CONTEXT_CREATION_FAILED, [65](#)
 - RT_ERROR_FILE_NOT_FOUND, [65](#)
 - RT_ERROR_ILLEGAL_SYMBOL, [65](#)
 - RT_ERROR_INSUFFICIENT_FREE_NODES, [65](#)
 - RT_ERROR_INVALID_CONTEXT, [65](#)
 - RT_ERROR_INVALID_DEVICE, [65](#)
 - RT_ERROR_INVALID_DRIVER_VERSION, [65](#)
 - RT_ERROR_INVALID_IMAGE, [65](#)
 - RT_ERROR_INVALID_SOURCE, [65](#)
 - RT_ERROR_INVALID_VALUE, [65](#)
 - RT_ERROR_LAUNCH_FAILED, [65](#)
 - RT_ERROR_MEMORY_ALLOCATION_FAILED,

- 65
- RT_ERROR_NETWORK_INIT_FAILED, 65
- RT_ERROR_NETWORK_LOAD_FAILED, 65
- RT_ERROR_NO_DEVICE, 65
- RT_ERROR_NOT_SUPPORTED, 65
- RT_ERROR_OBJECT_CREATION_FAILED, 65
- RT_ERROR_RESOURCE_ALREADY_REGISTERED, 65
- RT_ERROR_RESOURCE_NOT_REGISTERED, 65
- RT_ERROR_TYPE_MISMATCH, 65
- RT_ERROR_UNKNOWN, 65
- RT_ERROR_VARIABLE_NOT_FOUND, 65
- RT_ERROR_VARIABLE_REDECLARED, 65
- RT_ERROR_VERSION_MISMATCH, 65
- RT_EXCEPTION_ALL, 62
- RT_EXCEPTION_BUFFER_ID_INVALID, 62
- RT_EXCEPTION_BUFFER_INDEX_OUT_OF_BOUNDS, 62
- RT_EXCEPTION_INDEX_OUT_OF_BOUNDS, 62
- RT_EXCEPTION_INTERNAL_ERROR, 62
- RT_EXCEPTION_INVALID_RAY, 62
- RT_EXCEPTION_PROGRAM_ID_INVALID, 62
- RT_EXCEPTION_STACK_OVERFLOW, 62
- RT_EXCEPTION_TEXTURE_ID_INVALID, 62
- RT_EXCEPTION_USER, 62
- RT_FILTER_LINEAR, 62
- RT_FILTER_NEAREST, 62
- RT_FILTER_NONE, 62
- RT_FORMAT_BUFFER_ID, 63
- RT_FORMAT_BYTE, 62
- RT_FORMAT_BYTE2, 62
- RT_FORMAT_BYTE3, 62
- RT_FORMAT_BYTE4, 62
- RT_FORMAT_FLOAT, 62
- RT_FORMAT_FLOAT2, 62
- RT_FORMAT_FLOAT3, 62
- RT_FORMAT_FLOAT4, 62
- RT_FORMAT_HALF, 63
- RT_FORMAT_HALF2, 63
- RT_FORMAT_HALF3, 63
- RT_FORMAT_HALF4, 63
- RT_FORMAT_INT, 63
- RT_FORMAT_INT2, 63
- RT_FORMAT_INT3, 63
- RT_FORMAT_INT4, 63
- RT_FORMAT_PROGRAM_ID, 63
- RT_FORMAT_SHORT, 62
- RT_FORMAT_SHORT2, 62
- RT_FORMAT_SHORT3, 62
- RT_FORMAT_SHORT4, 62
- RT_FORMAT_UNKNOWN, 62
- RT_FORMAT_UNSIGNED_BYTE, 62
- RT_FORMAT_UNSIGNED_BYTE2, 62
- RT_FORMAT_UNSIGNED_BYTE3, 62
- RT_FORMAT_UNSIGNED_BYTE4, 62
- RT_FORMAT_UNSIGNED_INT, 63
- RT_FORMAT_UNSIGNED_INT2, 63
- RT_FORMAT_UNSIGNED_INT3, 63
- RT_FORMAT_UNSIGNED_INT4, 63
- RT_FORMAT_USER, 63
- RT_OBJECTTYPE_BUFFER, 63
- RT_OBJECTTYPE_FLOAT, 64
- RT_OBJECTTYPE_FLOAT2, 64
- RT_OBJECTTYPE_FLOAT3, 64
- RT_OBJECTTYPE_FLOAT4, 64
- RT_OBJECTTYPE_GEOMETRY_GROUP, 63
- RT_OBJECTTYPE_GEOMETRY_INSTANCE, 63
- RT_OBJECTTYPE_GROUP, 63
- RT_OBJECTTYPE_INT, 64
- RT_OBJECTTYPE_INT2, 64
- RT_OBJECTTYPE_INT3, 64
- RT_OBJECTTYPE_INT4, 64
- RT_OBJECTTYPE_MATRIX_FLOAT2x2, 64
- RT_OBJECTTYPE_MATRIX_FLOAT2x3, 64
- RT_OBJECTTYPE_MATRIX_FLOAT2x4, 64
- RT_OBJECTTYPE_MATRIX_FLOAT3x2, 64
- RT_OBJECTTYPE_MATRIX_FLOAT3x3, 64
- RT_OBJECTTYPE_MATRIX_FLOAT3x4, 64
- RT_OBJECTTYPE_MATRIX_FLOAT4x2, 64
- RT_OBJECTTYPE_MATRIX_FLOAT4x3, 64
- RT_OBJECTTYPE_MATRIX_FLOAT4x4, 64
- RT_OBJECTTYPE_OBJECT, 63
- RT_OBJECTTYPE_PROGRAM, 64
- RT_OBJECTTYPE_SELECTOR, 63
- RT_OBJECTTYPE_TEXTURE_SAMPLER, 63
- RT_OBJECTTYPE_TRANSFORM, 63
- RT_OBJECTTYPE_UNKNOWN, 63
- RT_OBJECTTYPE_UNSIGNED_INT, 64
- RT_OBJECTTYPE_UNSIGNED_INT2, 64
- RT_OBJECTTYPE_UNSIGNED_INT3, 64
- RT_OBJECTTYPE_UNSIGNED_INT4, 64
- RT_OBJECTTYPE_USER, 64
- RT_PROGRAM_ID_NULL, 64
- RT_REMOTEDEVICE_ATTRIBUTE_CLUSTER_URL, 64
- RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONS, 64
- RT_REMOTEDEVICE_ATTRIBUTE_GPU_TOTAL_MEMORY, 64
- RT_REMOTEDEVICE_ATTRIBUTE_HEAD_NODE_URL, 64
- RT_REMOTEDEVICE_ATTRIBUTE_NAME, 64
- RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS, 64
- RT_REMOTEDEVICE_ATTRIBUTE_NUM_FREE_NODES, 64
- RT_REMOTEDEVICE_ATTRIBUTE_NUM_GPUS, 64
- RT_REMOTEDEVICE_ATTRIBUTE_NUM_RESERVED_NODES, 64

- RT_REMOTEDEVICE_ATTRIBUTE_NUM_TOTAL_NODES, [64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_STATUS, [64](#)
- RT_REMOTEDEVICE_STATUS_CONNECTED, [65](#)
- RT_REMOTEDEVICE_STATUS_DISCONNECTED, [65](#)
- RT_REMOTEDEVICE_STATUS_READY, [65](#)
- RT_REMOTEDEVICE_STATUS_RESERVED, [65](#)
- RT_SUCCESS, [65](#)
- RT_TARGET_GL_RENDER_BUFFER, [63](#)
- RT_TARGET_GL_TEXTURE_1D, [63](#)
- RT_TARGET_GL_TEXTURE_1D_ARRAY, [63](#)
- RT_TARGET_GL_TEXTURE_2D, [63](#)
- RT_TARGET_GL_TEXTURE_2D_ARRAY, [63](#)
- RT_TARGET_GL_TEXTURE_3D, [63](#)
- RT_TARGET_GL_TEXTURE_CUBE_MAP, [63](#)
- RT_TARGET_GL_TEXTURE_CUBE_MAP_ARRAY, [63](#)
- RT_TARGET_GL_TEXTURE_RECTANGLE, [63](#)
- RT_TEXTURE_ID_NULL, [66](#)
- RT_TEXTURE_INDEX_ARRAY_INDEX, [66](#)
- RT_TEXTURE_INDEX_NORMALIZED_COORDINATES, [66](#)
- RT_TEXTURE_READ_ELEMENT_TYPE, [66](#)
- RT_TEXTURE_READ_ELEMENT_TYPE_SRGB, [66](#)
- RT_TEXTURE_READ_NORMALIZED_FLOAT, [66](#)
- RT_TEXTURE_READ_NORMALIZED_FLOAT_SRGB, [66](#)
- RT_TIMEOUT_CALLBACK, [65](#)
- RT_WRAP_CLAMP_TO_BORDER, [66](#)
- RT_WRAP_CLAMP_TO_EDGE, [66](#)
- RT_WRAP_MIRROR, [66](#)
- RT_WRAP_REPEAT, [66](#)
- optix_defines.h
 - RT_INTERNAL_INVERSE_TRANSPOSE, [67](#)
 - RT_OBJECT_TO_WORLD, [67](#)
 - RT_WORLD_TO_OBJECT, [67](#)
- optix_prime_declarations.h
 - RTP_BUFFER_FORMAT_HIT_BITMASK, [257](#)
 - RTP_BUFFER_FORMAT_HIT_T, [257](#)
 - RTP_BUFFER_FORMAT_HIT_T_TRIID, [257](#)
 - RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID, [257](#)
 - RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID_U_V, [257](#)
 - RTP_BUFFER_FORMAT_HIT_T_TRIID_U_V, [257](#)
 - RTP_BUFFER_FORMAT_INDICES_INT3, [256](#)
 - RTP_BUFFER_FORMAT_INDICES_INT3_MASK_INT, [256](#)
 - RTP_BUFFER_FORMAT_INSTANCE_MODEL, [257](#)
 - RTP_BUFFER_FORMAT_RAY_ORIGIN_DIRECTION, [256](#)
 - RTP_BUFFER_FORMAT_RAY_ORIGIN_MASK_DIRECTION_TMAX, [256](#)
 - RTP_BUFFER_FORMAT_RAY_ORIGIN_TMIN_DIRECTION_TMAX, [256](#)
 - RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x3, [257](#)
 - RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x4, [257](#)
 - RTP_BUFFER_FORMAT_VERTEX_FLOAT3, [256](#)
 - RTP_BUFFER_FORMAT_VERTEX_FLOAT4, [256](#)
 - RTP_BUFFER_TYPE_CUDA_LINEAR, [257](#)
 - RTP_BUFFER_TYPE_HOST, [257](#)
 - RTP_BUILDER_PARAM_CHUNK_SIZE, [257](#)
 - RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES, [257](#)
 - RTP_CONTEXT_TYPE_CPU, [257](#)
 - RTP_CONTEXT_TYPE_CUDA, [257](#)
 - RTP_ERROR_INVALID_CONTEXT, [258](#)
 - RTP_ERROR_INVALID_HANDLE, [258](#)
 - RTP_ERROR_INVALID_OPERATION, [258](#)
 - RTP_ERROR_INVALID_VALUE, [258](#)
 - RTP_ERROR_MEMORY_ALLOCATION_FAILED, [258](#)
 - RTP_ERROR_NOT_SUPPORTED, [258](#)
 - RTP_ERROR_OBJECT_CREATION_FAILED, [258](#)
 - RTP_ERROR_OUT_OF_MEMORY, [258](#)
 - RTP_ERROR_UNKNOWN, [258](#)
 - RTP_ERROR_VALIDATION_ERROR, [258](#)
 - RTP_MODEL_HINT_ASYNC, [257](#)
 - RTP_MODEL_HINT_MASK_UPDATE, [257](#)
 - RTP_MODEL_HINT_NONE, [257](#)
 - RTP_MODEL_HINT_USER_TRIANGLES_AFTER_COPY_SET, [257](#)
 - RTP_QUERY_HINT_ASYNC, [258](#)
 - RTP_QUERY_HINT_NONE, [258](#)
 - RTP_QUERY_HINT_WATERTIGHT, [258](#)
 - RTP_QUERY_TYPE_ANY, [258](#)
 - RTP_QUERY_TYPE_CLOSEST, [258](#)
 - RTP_SUCCESS, [258](#)
- optix::APIObj, [7](#)
 - checkError, [8](#)
- optix::Aabb, [3](#)
 - Aabb, [4](#)
 - area, [4](#)
 - center, [4](#)
 - contains, [4](#)
 - distance, [4](#)
 - distance2, [4](#)
 - enlarge, [4](#)
 - extent, [4](#)
 - halfArea, [4](#)
 - include, [4, 5](#)
 - intersection, [5](#)
 - intersects, [5](#)
 - invalidate, [5](#)
 - isFlat, [5](#)
 - longestAxis, [5](#)
 - m_max, [6](#)
 - m_min, [6](#)

- maxExtent, 5
- operator==, 5
- set, 5
- signedDistance, 5
- valid, 5
- volume, 6
- optix::AccelerationObj, 6
 - getDataSize, 7
 - getProperty, 7
 - markDirty, 7
 - setProperty, 7
- optix::BufferObj, 11
 - bindProgressiveStream, 12
 - getDevicePointer, 12
 - getGLBOld, 12
 - getId, 12
 - map, 12
 - setAttribute, 13
 - setFormat, 13
 - setSize, 13
- optix::ContextObj, 14
 - checkError, 17
 - create1DLayeredBuffer, 17
 - create2DLayeredBuffer, 17
 - createAcceleration, 17
 - createBuffer, 17, 18
 - createBufferForCUDA, 18
 - createCubeBuffer, 18
 - createCubeLayeredBuffer, 18
 - createGeometryGroup, 18
 - createGeometryInstance, 18
 - createGroup, 18
 - createMipmappedBuffer, 18, 19
 - declareVariable, 19
 - getBufferFromId, 19
 - getContext, 19
 - getEnabledDeviceCount, 19
 - getMaxTextureCount, 19
 - getProgramFromId, 19
 - getTextureSamplerFromId, 19
 - getVariableCount, 19
 - launch, 19
 - launchProgressive, 20
 - queryVariable, 20
 - setCPUNumThreads, 20
 - setDevices, 20
 - setPrintEnabled, 20
 - setRayGenerationProgram, 20
 - setStackSize, 20
 - setTimeoutCallback, 20
- optix::DestroyableObj, 20
- optix::Exception, 22
 - ~Exception, 22
 - makeException, 22
- optix::GeometryGroupObj, 23
 - removeChild, 23
 - setAcceleration, 24
 - setChildCount, 24
- optix::GeometryInstanceObj, 24
 - declareVariable, 25
 - getVariableCount, 25
 - queryVariable, 25
 - setGeometry, 25
- optix::GeometryObj, 25
 - declareVariable, 26
 - getPrimitiveCount, 26
 - getPrimitiveIndexOffset, 26
 - getVariableCount, 26
 - markDirty, 26
 - queryVariable, 26
 - setBoundingBoxProgram, 26
 - setPrimitiveCount, 27
 - setPrimitiveIndexOffset, 27
- optix::GroupObj, 27
 - removeChild, 28
 - setAcceleration, 28
 - setChildCount, 28
- optix::Handle
 - take, 29
- optix::Handle< T >, 28
- optix::MaterialObj, 30
 - declareVariable, 30
 - getVariableCount, 30
 - queryVariable, 30
 - setClosestHitProgram, 31
- optix::Matrix
 - det, 32
 - fromBasis, 32
 - getCol, 32
 - getData, 32, 33
 - getRow, 33
 - identity, 33
 - inverse, 33
 - Matrix, 32
 - operator<, 33
 - operator=, 33
 - rotate, 33
 - scale, 33
 - setCol, 33
 - setRow, 33
 - translate, 34
 - transpose, 34
- optix::Matrix< M, N >, 31
- optix::Onb, 35
- optix::ProgramObj, 36
 - declareVariable, 36
 - getId, 37
 - getVariableCount, 37
 - queryVariable, 37
- optix::RemoteDeviceObj, 39
- optix::ScopedObj, 41
 - declareVariable, 42
 - getVariableCount, 42
 - queryVariable, 42
- optix::SelectorObj, 42
 - removeChild, 43, 44

- setChildCount, [44](#)
 - setVisitProgram, [44](#)
- optix::TextureSamplerObj, [44](#)
 - getId, [45](#)
 - registerGLTexture, [45](#)
 - setBuffer, [45](#)
 - setMipLevelCount, [45](#)
- optix::TransformObj, [45](#)
 - setChild, [46](#)
 - setMatrix, [46](#)
- optix::VariableObj, [46](#)
- optix::VectorDim< 2 >, [49](#)
- optix::VectorDim< 3 >, [49](#)
- optix::VectorDim< 4 >, [49](#)
- optix::VectorDim< DIM >, [49](#)
- optix::VectorTypes< float, 1 >, [49](#)
- optix::VectorTypes< float, 2 >, [50](#)
- optix::VectorTypes< float, 3 >, [50](#)
- optix::VectorTypes< float, 4 >, [50](#)
- optix::VectorTypes< int, 1 >, [50](#)
- optix::VectorTypes< int, 2 >, [50](#)
- optix::VectorTypes< int, 3 >, [51](#)
- optix::VectorTypes< int, 4 >, [51](#)
- optix::VectorTypes< T, Dim >, [49](#)
- optix::VectorTypes< unsigned int, 1 >, [51](#)
- optix::VectorTypes< unsigned int, 2 >, [51](#)
- optix::VectorTypes< unsigned int, 3 >, [52](#)
- optix::VectorTypes< unsigned int, 4 >, [52](#)
- optix::boundCallableProgramId< T >, [8](#)
- optix::buffer< T, Dim >, [8](#)
- optix::buffer< T, Dim >::type< T2 >, [46](#)
- optix::bufferId< T, Dim >, [10](#)
- optix::callableProgramId< T >, [13](#)
- optix::prime::BufferDescObj, [9](#)
- optix::prime::ContextObj, [14](#)
- optix::prime::Exception, [21](#)
- optix::prime::ModelObj, [34](#)
 - setBuilderParameter, [35](#)
 - setInstances, [35](#)
 - setTriangles, [35](#)
- optix::prime::QueryObj, [37](#)
- optix_cuda_interop.h, [53](#)
 - rtBufferCreateForCUDA, [53](#)
 - rtBufferGetDevicePointer, [53](#)
 - rtBufferMarkDirty, [54](#)
 - rtBufferSetDevicePointer, [54](#)
- optix_datatypes.h, [55](#)
 - RT_DEFAULT_MAX, [55](#)
- optix_declarations.h, [56](#)
 - RTbufferattribute, [60](#)
 - RTbufferflag, [60](#)
 - RTbufferidnull, [60](#)
 - RTbuffermapflag, [60](#)
 - RTbuffertype, [61](#)
 - RTcontextattribute, [61](#)
 - RTdeviceattribute, [61](#)
 - RTexception, [61](#)
 - RTfiltermode, [62](#)
 - RTformat, [62](#)
 - RTgltarget, [63](#)
 - RTobjecttype, [63](#)
 - RTprogramidnull, [64](#)
 - RTremotedeviceattribute, [64](#)
 - RTremotedevicestatus, [64](#)
 - RTresult, [65](#)
 - RTtextureidnull, [65](#)
 - RTtextureindexmode, [66](#)
 - RTtexturereadmode, [66](#)
 - RTwrapmode, [66](#)
- optix_defines.h, [66](#)
 - RTtransformflags, [67](#)
 - RTtransformkind, [67](#)
- optix_device.h, [67](#)
 - RT_PROGRAM, [74](#)
 - rtBuffer, [75](#)
 - rtBufferId, [75](#)
 - rtCallableProgram, [75](#)
 - rtCallableProgramId, [76](#)
 - rtCallableProgramX, [76](#)
 - rtDeclareAnnotation, [77](#)
 - rtDeclareVariable, [78](#)
 - rtGetExceptionCode, [79](#)
 - rtGetTransform, [79](#)
 - rtIgnoreIntersection, [80](#)
 - rtIntersectChild, [80](#)
 - rtPotentialIntersection, [81](#)
 - rtPrintExceptionDetails, [81](#)
 - rtPrintf, [81–85](#)
 - rtReportIntersection, [86](#)
 - rtTerminateRay, [86](#)
 - rtTextureSampler, [79](#)
 - rtThrow, [86](#)
 - rtTrace, [87](#)
 - rtTransformNormal, [87](#)
 - rtTransformPoint, [87](#)
 - rtTransformVector, [88](#)
- optix_gl_interop.h, [88](#)
 - rtBufferCreateFromGLBO, [89](#)
 - rtBufferGLRegister, [90](#)
 - rtBufferGLUnregister, [91](#)
 - rtBufferGetGLBOId, [90](#)
 - rtDeviceGetWGLDevice, [91](#)
 - rtTextureSamplerCreateFromGLImage, [92](#)
 - rtTextureSamplerGLRegister, [93](#)
 - rtTextureSamplerGLUnregister, [94](#)
 - rtTextureSamplerGetGLImageId, [93](#)
- optix_host.h, [95](#)
 - RTacceleration, [102](#)
 - RTbuffer, [102](#)
 - RTcontext, [102](#)
 - RTgeometry, [102](#)
 - RTgeometrygroup, [102](#)
 - RTgeometryinstance, [102](#)
 - RTgroup, [102](#)
 - RTmaterial, [102](#)
 - RTobject, [103](#)

- RTprogram, 103
- RTremotedevice, 103
- RTselector, 103
- RTtexturesampler, 103
- RTtimeoutcallback, 103
- RTtransform, 103
- RTvariable, 103
- rtAccelerationCreate, 103
- rtAccelerationDestroy, 104
- rtAccelerationGetBuilder, 104
- rtAccelerationGetContext, 105
- rtAccelerationGetData, 105
- rtAccelerationGetDataSize, 105
- rtAccelerationGetProperty, 105
- rtAccelerationGetTraverser, 106
- rtAccelerationIsDirty, 106
- rtAccelerationMarkDirty, 106
- rtAccelerationSetBuilder, 107
- rtAccelerationSetData, 108
- rtAccelerationSetProperty, 108
- rtAccelerationSetTraverser, 109
- rtAccelerationValidate, 109
- rtBufferBindProgressiveStream, 109
- rtBufferCreate, 110
- rtBufferDestroy, 111
- rtBufferGetAttribute, 111
- rtBufferGetContext, 112
- rtBufferGetDimensionality, 112
- rtBufferGetElementSize, 113
- rtBufferGetFormat, 113
- rtBufferGetId, 114
- rtBufferGetMipLevelCount, 114
- rtBufferGetMipLevelSize1D, 115
- rtBufferGetMipLevelSize2D, 115
- rtBufferGetMipLevelSize3D, 116
- rtBufferGetProgressiveUpdateReady, 116
- rtBufferGetSize1D, 117
- rtBufferGetSize2D, 117
- rtBufferGetSize3D, 118
- rtBufferGetSizev, 118
- rtBufferMap, 119
- rtBufferMapEx, 119
- rtBufferSetAttribute, 120
- rtBufferSetElementSize, 121
- rtBufferSetFormat, 121
- rtBufferSetMipLevelCount, 123
- rtBufferSetSize1D, 123
- rtBufferSetSize2D, 124
- rtBufferSetSize3D, 124
- rtBufferSetSizev, 125
- rtBufferUnmap, 126
- rtBufferUnmapEx, 126
- rtBufferValidate, 127
- rtContextCompile, 127
- rtContextCreate, 127
- rtContextDeclareVariable, 128
- rtContextDestroy, 128
- rtContextGetAttribute, 129
- rtContextGetBufferFromId, 130
- rtContextGetDeviceCount, 130
- rtContextGetDevices, 131
- rtContextGetEntryPointCount, 131
- rtContextGetErrorString, 131
- rtContextGetExceptionEnabled, 132
- rtContextGetExceptionProgram, 132
- rtContextGetMissProgram, 133
- rtContextGetPrintBufferSize, 133
- rtContextGetPrintEnabled, 134
- rtContextGetPrintLaunchIndex, 134
- rtContextGetProgramFromId, 135
- rtContextGetRayGenerationProgram, 135
- rtContextGetRayTypeCount, 136
- rtContextGetRunningState, 136
- rtContextGetStackSize, 136
- rtContextGetTextureSamplerFromId, 137
- rtContextGetVariable, 137
- rtContextGetVariableCount, 138
- rtContextLaunch1D, 138
- rtContextLaunch2D, 139
- rtContextLaunch3D, 139
- rtContextLaunchProgressive2D, 139
- rtContextQueryVariable, 140
- rtContextRemoveVariable, 141
- rtContextSetAttribute, 141
- rtContextSetDevices, 142
- rtContextSetEntryPointCount, 142
- rtContextSetExceptionEnabled, 142
- rtContextSetExceptionProgram, 144
- rtContextSetMissProgram, 144
- rtContextSetPrintBufferSize, 145
- rtContextSetPrintEnabled, 145
- rtContextSetPrintLaunchIndex, 145
- rtContextSetRayGenerationProgram, 147
- rtContextSetRayTypeCount, 147
- rtContextSetRemoteDevice, 149
- rtContextSetStackSize, 149
- rtContextSetTimeoutCallback, 150
- rtContextStopProgressive, 150
- rtContextValidate, 151
- rtDeviceGetAttribute, 151
- rtDeviceGetDeviceCount, 152
- rtGeometryCreate, 153
- rtGeometryDeclareVariable, 153
- rtGeometryDestroy, 154
- rtGeometryGetBoundingBoxProgram, 154
- rtGeometryGetContext, 155
- rtGeometryGetIntersectionProgram, 155
- rtGeometryGetPrimitiveCount, 156
- rtGeometryGetPrimitiveIndexOffset, 156
- rtGeometryGetVariable, 157
- rtGeometryGetVariableCount, 157
- rtGeometryGroupCreate, 158
- rtGeometryGroupDestroy, 158
- rtGeometryGroupGetAcceleration, 159
- rtGeometryGroupGetChild, 159
- rtGeometryGroupGetChildCount, 160

rtGeometryGroupGetContext, 160
rtGeometryGroupSetAcceleration, 161
rtGeometryGroupSetChild, 161
rtGeometryGroupSetChildCount, 163
rtGeometryGroupValidate, 163
rtGeometryInstanceCreate, 164
rtGeometryInstanceDeclareVariable, 164
rtGeometryInstanceDestroy, 166
rtGeometryInstanceGetContext, 166
rtGeometryInstanceGetGeometry, 167
rtGeometryInstanceGetMaterial, 167
rtGeometryInstanceGetMaterialCount, 168
rtGeometryInstanceGetVariable, 168
rtGeometryInstanceGetVariableCount, 169
rtGeometryInstanceQueryVariable, 169
rtGeometryInstanceRemoveVariable, 170
rtGeometryInstanceSetGeometry, 170
rtGeometryInstanceSetMaterial, 171
rtGeometryInstanceSetMaterialCount, 171
rtGeometryInstanceValidate, 172
rtGeometryIsDirty, 172
rtGeometryMarkDirty, 172
rtGeometryQueryVariable, 173
rtGeometryRemoveVariable, 173
rtGeometrySetBoundingBoxProgram, 174
rtGeometrySetIntersectionProgram, 174
rtGeometrySetPrimitiveCount, 175
rtGeometrySetPrimitiveIndexOffset, 175
rtGeometryValidate, 176
rtGetVersion, 176
rtGroupCreate, 176
rtGroupDestroy, 178
rtGroupGetAcceleration, 178
rtGroupGetChild, 179
rtGroupGetChildCount, 179
rtGroupGetChildType, 179
rtGroupGetContext, 180
rtGroupSetAcceleration, 180
rtGroupSetChild, 181
rtGroupSetChildCount, 181
rtGroupValidate, 182
rtMaterialCreate, 182
rtMaterialDeclareVariable, 183
rtMaterialDestroy, 183
rtMaterialGetAnyHitProgram, 184
rtMaterialGetClosestHitProgram, 184
rtMaterialGetContext, 185
rtMaterialGetVariable, 185
rtMaterialGetVariableCount, 186
rtMaterialQueryVariable, 186
rtMaterialRemoveVariable, 187
rtMaterialSetAnyHitProgram, 187
rtMaterialSetClosestHitProgram, 189
rtMaterialValidate, 189
rtProgramCreateFromPTXFile, 190
rtProgramCreateFromPTXString, 190
rtProgramDeclareVariable, 191
rtProgramDestroy, 192
rtProgramGetContext, 192
rtProgramGetId, 192
rtProgramGetVariable, 193
rtProgramGetVariableCount, 193
rtProgramQueryVariable, 194
rtProgramRemoveVariable, 194
rtProgramValidate, 195
rtRemoteDeviceCreate, 195
rtRemoteDeviceDestroy, 196
rtRemoteDeviceGetAttribute, 196
rtRemoteDeviceRelease, 198
rtRemoteDeviceReserve, 199
rtSelectorCreate, 199
rtSelectorDeclareVariable, 200
rtSelectorDestroy, 200
rtSelectorGetChild, 202
rtSelectorGetChildCount, 202
rtSelectorGetChildType, 203
rtSelectorGetContext, 203
rtSelectorGetVariable, 204
rtSelectorGetVariableCount, 204
rtSelectorGetVisitProgram, 205
rtSelectorQueryVariable, 205
rtSelectorRemoveVariable, 206
rtSelectorSetChild, 206
rtSelectorSetChildCount, 207
rtSelectorSetVisitProgram, 207
rtSelectorValidate, 208
rtTextureSamplerCreate, 208
rtTextureSamplerDestroy, 209
rtTextureSamplerGetArraySize, 209
rtTextureSamplerGetBuffer, 209
rtTextureSamplerGetContext, 210
rtTextureSamplerGetFilteringModes, 210
rtTextureSamplerGetId, 211
rtTextureSamplerGetIndexingMode, 211
rtTextureSamplerGetMaxAnisotropy, 212
rtTextureSamplerGetMipLevelBias, 212
rtTextureSamplerGetMipLevelClamp, 213
rtTextureSamplerGetMipLevelCount, 213
rtTextureSamplerGetReadMode, 213
rtTextureSamplerGetWrapMode, 214
rtTextureSamplerSetArraySize, 214
rtTextureSamplerSetBuffer, 214
rtTextureSamplerSetFilteringModes, 215
rtTextureSamplerSetIndexingMode, 215
rtTextureSamplerSetMaxAnisotropy, 216
rtTextureSamplerSetMipLevelBias, 216
rtTextureSamplerSetMipLevelClamp, 217
rtTextureSamplerSetMipLevelCount, 217
rtTextureSamplerSetReadMode, 217
rtTextureSamplerSetWrapMode, 218
rtTextureSamplerValidate, 219
rtTransformCreate, 219
rtTransformDestroy, 220
rtTransformGetChild, 220
rtTransformGetChildType, 221
rtTransformGetContext, 221

rtTransformGetMatrix, [222](#)
rtTransformSetChild, [222](#)
rtTransformSetMatrix, [223](#)
rtTransformValidate, [224](#)
rtVariableGet1f, [224](#)
rtVariableGet1fv, [225](#)
rtVariableGet1i, [225](#)
rtVariableGet1iv, [225](#)
rtVariableGet1ui, [225](#)
rtVariableGet1uiv, [225](#)
rtVariableGet2f, [225](#)
rtVariableGet2fv, [226](#)
rtVariableGet2i, [226](#)
rtVariableGet2iv, [226](#)
rtVariableGet2ui, [226](#)
rtVariableGet2uiv, [226](#)
rtVariableGet3f, [226](#)
rtVariableGet3fv, [226](#)
rtVariableGet3i, [227](#)
rtVariableGet3iv, [227](#)
rtVariableGet3ui, [227](#)
rtVariableGet3uiv, [227](#)
rtVariableGet4f, [227](#)
rtVariableGet4fv, [227](#)
rtVariableGet4i, [228](#)
rtVariableGet4iv, [228](#)
rtVariableGet4ui, [228](#)
rtVariableGet4uiv, [228](#)
rtVariableGetAnnotation, [228](#)
rtVariableGetContext, [229](#)
rtVariableGetMatrix2x2fv, [229](#)
rtVariableGetMatrix2x3fv, [229](#)
rtVariableGetMatrix2x4fv, [229](#)
rtVariableGetMatrix3x2fv, [230](#)
rtVariableGetMatrix3x3fv, [230](#)
rtVariableGetMatrix3x4fv, [230](#)
rtVariableGetMatrix4x2fv, [230](#)
rtVariableGetMatrix4x3fv, [230](#)
rtVariableGetMatrix4x4fv, [230](#)
rtVariableGetName, [230](#)
rtVariableGetObject, [231](#)
rtVariableGetSize, [231](#)
rtVariableGetType, [232](#)
rtVariableGetUserData, [233](#)
rtVariableSet1f, [234](#)
rtVariableSet1fv, [234](#)
rtVariableSet1i, [235](#)
rtVariableSet1iv, [235](#)
rtVariableSet1ui, [235](#)
rtVariableSet1uiv, [235](#)
rtVariableSet2f, [235](#)
rtVariableSet2fv, [235](#)
rtVariableSet2i, [235](#)
rtVariableSet2iv, [236](#)
rtVariableSet2ui, [236](#)
rtVariableSet2uiv, [236](#)
rtVariableSet3f, [236](#)
rtVariableSet3fv, [236](#)
rtVariableSet3i, [236](#)
rtVariableSet3iv, [236](#)
rtVariableSet3ui, [237](#)
rtVariableSet3uiv, [237](#)
rtVariableSet4f, [237](#)
rtVariableSet4fv, [237](#)
rtVariableSet4i, [237](#)
rtVariableSet4iv, [237](#)
rtVariableSet4ui, [237](#)
rtVariableSet4uiv, [238](#)
rtVariableSetMatrix2x2fv, [238](#)
rtVariableSetMatrix2x3fv, [238](#)
rtVariableSetMatrix2x4fv, [238](#)
rtVariableSetMatrix3x2fv, [238](#)
rtVariableSetMatrix3x3fv, [238](#)
rtVariableSetMatrix3x4fv, [239](#)
rtVariableSetMatrix4x2fv, [239](#)
rtVariableSetMatrix4x3fv, [239](#)
rtVariableSetMatrix4x4fv, [239](#)
rtVariableSetObject, [239](#)
rtVariableSetUserData, [240](#)
optix_prime.h, [240](#)
RTPbufferdesc, [242](#)
RTPcontext, [242](#)
RTPmodel, [242](#)
RTPquery, [242](#)
rtpBufferDescCreate, [242](#)
rtpBufferDescDestroy, [242](#)
rtpBufferDescGetContext, [243](#)
rtpBufferDescSetCudaDeviceNumber, [243](#)
rtpBufferDescSetRange, [243](#)
rtpBufferDescSetStride, [244](#)
rtpContextCreate, [244](#)
rtpContextDestroy, [245](#)
rtpContextGetLastErrorString, [245](#)
rtpContextSetCpuThreads, [245](#)
rtpContextSetCudaDeviceNumbers, [246](#)
rtpGetErrorString, [246](#)
rtpGetVersion, [247](#)
rtpGetVersionString, [247](#)
rtpHostBufferLock, [247](#)
rtpHostBufferUnlock, [247](#)
rtpModelCopy, [248](#)
rtpModelCreate, [248](#)
rtpModelDestroy, [248](#)
rtpModelFinish, [249](#)
rtpModelGetContext, [249](#)
rtpModelGetFinished, [249](#)
rtpModelSetBuilderParameter, [250](#)
rtpModelSetInstances, [250](#)
rtpModelSetTriangles, [251](#)
rtpModelUpdate, [251](#)
rtpQueryCreate, [252](#)
rtpQueryDestroy, [252](#)
rtpQueryExecute, [253](#)
rtpQueryFinish, [253](#)
rtpQueryGetContext, [253](#)
rtpQueryGetFinished, [254](#)

- rtpQuerySetCudaStream, [254](#)
 - rtpQuerySetHits, [254](#)
 - rtpQuerySetRays, [255](#)
- optix_prime_declarations.h, [255](#)
 - RTPbufferformat, [256](#)
 - RTPbuffertype, [257](#)
 - RTPbuilderparam, [257](#)
 - RTPcontexttype, [257](#)
 - RTPmodelhint, [257](#)
 - RTPqueryhint, [257](#)
 - RTPquerytype, [258](#)
 - RTPresult, [258](#)
- optix_primecpp.h, [258](#)
- optix_world.h, [259](#)
- optixpp_namespace.h, [259](#)
- optixu.h, [262](#)
 - RTU_CHECK_ERROR, [263](#)
 - rtuCUDACompileFile, [265](#)
 - rtuCUDACompileString, [265](#)
 - rtuCUDAGetCompileResult, [265](#)
 - rtuCreateClusteredMesh, [263](#)
 - rtuCreateClusteredMeshExt, [264](#)
 - rtuGeometryGroupAddChild, [266](#)
 - rtuGeometryGroupGetChildIndex, [266](#)
 - rtuGeometryGroupRemoveChild, [266](#)
 - rtuGeometryGroupRemoveChildByIndex, [266](#)
 - rtuGetSizeForRTformat, [266](#)
 - rtuGroupAddChild, [266](#)
 - rtuGroupGetChildIndex, [266](#)
 - rtuGroupRemoveChild, [266](#)
 - rtuGroupRemoveChildByIndex, [266](#)
 - rtuNameForType, [266](#)
 - rtuSelectorAddChild, [268](#)
 - rtuSelectorGetChildIndex, [268](#)
 - rtuSelectorRemoveChild, [268](#)
 - rtuSelectorRemoveChildByIndex, [268](#)
 - rtuTransformGetChild, [268](#)
 - rtuTransformGetChildType, [268](#)
 - rtuTransformSetChild, [268](#)
- optixu_traversal.h
 - RTU_INITOPTION_CPU_ONLY, [281](#)
 - RTU_INITOPTION_CULL_BACKFACE, [281](#)
 - RTU_INITOPTION_GPU_ONLY, [281](#)
 - RTU_INITOPTION_NONE, [281](#)
 - RTU_OPTION_INT_NUM_THREADS, [281](#)
 - RTU_OUTPUT_BACKFACING, [281](#)
 - RTU_OUTPUT_BARYCENTRIC, [281](#)
 - RTU_OUTPUT_NONE, [281](#)
 - RTU_OUTPUT_NORMAL, [281](#)
 - RTU_QUERY_TYPE_ANY_HIT, [281](#)
 - RTU_QUERY_TYPE_CLOSEST_HIT, [281](#)
 - RTU_QUERY_TYPE_COUNT, [281](#)
 - RTU_RAYFORMAT_COUNT, [282](#)
 - RTU_RAYFORMAT_ORIGIN_DIRECTION_INTERLEAVED, [282](#)
 - RTU_RAYFORMAT_ORIGIN_DIRECTION_TMIN_TMAX_INTERLEAVED, [282](#)
 - RTU_TRIFORMAT_COUNT, [282](#)
 - RTU_TRIFORMAT_MESH, [282](#)
 - RTU_TRIFORMAT_TRIANGLE_SOUP, [282](#)
- optixu_aabb_namespace.h, [268](#)
- optixu_math_namespace.h, [269](#)
- optixu_math_stream_namespace.h, [277](#)
- optixu_matrix_namespace.h, [278](#)
- optixu_traversal.h, [279](#)
 - RTUinitoptions, [281](#)
 - RTUoption, [281](#)
 - RTUoutput, [281](#)
 - RTUquerytype, [281](#)
 - RTUrayformat, [281](#)
 - RTUtraversal, [280](#)
 - RTUtriformat, [282](#)
 - rtuTraversalCreate, [282](#)
 - rtuTraversalDestroy, [282](#)
 - rtuTraversalGetAccelData, [282](#)
 - rtuTraversalGetAccelDataSize, [283](#)
 - rtuTraversalGetErrorString, [283](#)
 - rtuTraversalMapOutput, [283](#)
 - rtuTraversalMapRays, [283](#)
 - rtuTraversalMapResults, [283](#)
 - rtuTraversalPreprocess, [284](#)
 - rtuTraversalSetAccelData, [284](#)
 - rtuTraversalSetMesh, [284](#)
 - rtuTraversalSetOption, [284](#)
 - rtuTraversalSetTriangles, [284](#)
 - rtuTraversalTraverse, [285](#)
 - rtuTraversalUnmapOutput, [285](#)
 - rtuTraversalUnmapRays, [285](#)
 - rtuTraversalUnmapResults, [285](#)
- origin
 - Ray, [39](#)
- prim_id
 - RTUtraversalresult, [41](#)
- queryVariable
 - optix::ContextObj, [20](#)
 - optix::GeometryInstanceObj, [25](#)
 - optix::GeometryObj, [26](#)
 - optix::MaterialObj, [30](#)
 - optix::ProgramObj, [37](#)
 - optix::ScopedObj, [42](#)
- RT_BUFFER_ATTRIBUTE_STREAM_BITRATE
 - optix_declarations.h, [60](#)
- RT_BUFFER_ATTRIBUTE_STREAM_FORMAT
 - optix_declarations.h, [60](#)
- RT_BUFFER_ATTRIBUTE_STREAM_FPS
 - optix_declarations.h, [60](#)
- RT_BUFFER_ATTRIBUTE_STREAM_GAMMA
 - optix_declarations.h, [60](#)
- RT_BUFFER_COPY_ON_DIRTY
 - optix_declarations.h, [60](#)
- RT_BUFFER_CUBEMAP
 - optix_declarations.h, [60](#)
- RT_BUFFER_GPU_LOCAL
 - optix_declarations.h, [60](#)

RT_BUFFER_ID_NULL
 optix_declarations.h, 61

RT_BUFFER_INPUT
 optix_declarations.h, 61

RT_BUFFER_INPUT_OUTPUT
 optix_declarations.h, 61

RT_BUFFER_LAYERED
 optix_declarations.h, 61

RT_BUFFER_MAP_READ
 optix_declarations.h, 61

RT_BUFFER_MAP_READ_WRITE
 optix_declarations.h, 61

RT_BUFFER_MAP_WRITE
 optix_declarations.h, 61

RT_BUFFER_MAP_WRITE_DISCARD
 optix_declarations.h, 61

RT_BUFFER_OUTPUT
 optix_declarations.h, 61

RT_BUFFER_PROGRESSIVE_STREAM
 optix_declarations.h, 61

RT_CONTEXT_ATTRIBUTE_AVAILABLE_DEVICE_M-
 MEMORY
 optix_declarations.h, 61

RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS
 optix_declarations.h, 61

RT_CONTEXT_ATTRIBUTE_GPU_PAGING_ACTIVE
 optix_declarations.h, 61

RT_CONTEXT_ATTRIBUTE_GPU_PAGING_FORCE-
 D_OFF
 optix_declarations.h, 61

RT_CONTEXT_ATTRIBUTE_MAX_TEXTURE_COUN-
 T
 optix_declarations.h, 61

RT_CONTEXT_ATTRIBUTE_USED_HOST_MEMORY
 optix_declarations.h, 61

RT_DEVICE_ATTRIBUTE_CLOCK_RATE
 optix_declarations.h, 61

RT_DEVICE_ATTRIBUTE_COMPUTE_CAPABILITY
 optix_declarations.h, 61

RT_DEVICE_ATTRIBUTE_CUDA_DEVICE_ORDINAL
 optix_declarations.h, 61

RT_DEVICE_ATTRIBUTE_EXECUTION_TIMEOUT_E-
 NABLED
 optix_declarations.h, 61

RT_DEVICE_ATTRIBUTE_MAX_HARDWARE_TEXT-
 URE_COUNT
 optix_declarations.h, 61

RT_DEVICE_ATTRIBUTE_MAX_THREADS_PER_BL-
 OCK
 optix_declarations.h, 61

RT_DEVICE_ATTRIBUTE_MULTIPROCESSOR_CO-
 UNT
 optix_declarations.h, 61

RT_DEVICE_ATTRIBUTE_NAME
 optix_declarations.h, 61

RT_DEVICE_ATTRIBUTE_TCC_DRIVER
 optix_declarations.h, 61

RT_DEVICE_ATTRIBUTE_TOTAL_MEMORY
 optix_declarations.h, 61

RT_ERROR_ALREADY_MAPPED
 optix_declarations.h, 65

RT_ERROR_AUTHENTICATION_FAILED
 optix_declarations.h, 65

RT_ERROR_CLUSTER_ALREADY_RUNNING
 optix_declarations.h, 65

RT_ERROR_CLUSTER_NOT_RUNNING
 optix_declarations.h, 65

RT_ERROR_CONNECTION_ALREADY_EXISTS
 optix_declarations.h, 65

RT_ERROR_CONNECTION_FAILED
 optix_declarations.h, 65

RT_ERROR_CONTEXT_CREATION_FAILED
 optix_declarations.h, 65

RT_ERROR_FILE_NOT_FOUND
 optix_declarations.h, 65

RT_ERROR_ILLEGAL_SYMBOL
 optix_declarations.h, 65

RT_ERROR_INSUFFICIENT_FREE_NODES
 optix_declarations.h, 65

RT_ERROR_INVALID_CONTEXT
 optix_declarations.h, 65

RT_ERROR_INVALID_DEVICE
 optix_declarations.h, 65

RT_ERROR_INVALID_DRIVER_VERSION
 optix_declarations.h, 65

RT_ERROR_INVALID_IMAGE
 optix_declarations.h, 65

RT_ERROR_INVALID_SOURCE
 optix_declarations.h, 65

RT_ERROR_INVALID_VALUE
 optix_declarations.h, 65

RT_ERROR_LAUNCH_FAILED
 optix_declarations.h, 65

RT_ERROR_MEMORY_ALLOCATION_FAILED
 optix_declarations.h, 65

RT_ERROR_NETWORK_INIT_FAILED
 optix_declarations.h, 65

RT_ERROR_NETWORK_LOAD_FAILED
 optix_declarations.h, 65

RT_ERROR_NO_DEVICE
 optix_declarations.h, 65

RT_ERROR_NOT_SUPPORTED
 optix_declarations.h, 65

RT_ERROR_OBJECT_CREATION_FAILED
 optix_declarations.h, 65

RT_ERROR_RESOURCE_ALREADY_REGISTERED
 optix_declarations.h, 65

RT_ERROR_RESOURCE_NOT_REGISTERED
 optix_declarations.h, 65

RT_ERROR_TYPE_MISMATCH
 optix_declarations.h, 65

RT_ERROR_UNKNOWN
 optix_declarations.h, 65

RT_ERROR_VARIABLE_NOT_FOUND
 optix_declarations.h, 65

RT_ERROR_VARIABLE_REDECLARED

- [optix_declarations.h](#), [65](#)
- [RT_ERROR_VERSION_MISMATCH](#)
 - [optix_declarations.h](#), [65](#)
- [RT_EXCEPTION_ALL](#)
 - [optix_declarations.h](#), [62](#)
- [RT_EXCEPTION_BUFFER_ID_INVALID](#)
 - [optix_declarations.h](#), [62](#)
- [RT_EXCEPTION_BUFFER_INDEX_OUT_OF_BOUNDS](#)
 - [optix_declarations.h](#), [62](#)
- [RT_EXCEPTION_INDEX_OUT_OF_BOUNDS](#)
 - [optix_declarations.h](#), [62](#)
- [RT_EXCEPTION_INTERNAL_ERROR](#)
 - [optix_declarations.h](#), [62](#)
- [RT_EXCEPTION_INVALID_RAY](#)
 - [optix_declarations.h](#), [62](#)
- [RT_EXCEPTION_PROGRAM_ID_INVALID](#)
 - [optix_declarations.h](#), [62](#)
- [RT_EXCEPTION_STACK_OVERFLOW](#)
 - [optix_declarations.h](#), [62](#)
- [RT_EXCEPTION_TEXTURE_ID_INVALID](#)
 - [optix_declarations.h](#), [62](#)
- [RT_EXCEPTION_USER](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FILTER_LINEAR](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FILTER_NEAREST](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FILTER_NONE](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_BUFFER_ID](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_BYTE](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_BYTE2](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_BYTE3](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_BYTE4](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_FLOAT](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_FLOAT2](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_FLOAT3](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_FLOAT4](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_HALF](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_HALF2](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_HALF3](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_HALF4](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_INT](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_INT2](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_INT3](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_INT4](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_PROGRAM_ID](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_SHORT](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_SHORT2](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_SHORT3](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_SHORT4](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_UNKNOWN](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_UNSIGNED_BYTE](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_UNSIGNED_BYTE2](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_UNSIGNED_BYTE3](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_UNSIGNED_BYTE4](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_UNSIGNED_INT](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_UNSIGNED_INT2](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_UNSIGNED_INT3](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_UNSIGNED_INT4](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_UNSIGNED_SHORT](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_UNSIGNED_SHORT2](#)
 - [optix_declarations.h](#), [62](#)
- [RT_FORMAT_UNSIGNED_SHORT3](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_UNSIGNED_SHORT4](#)
 - [optix_declarations.h](#), [63](#)
- [RT_FORMAT_USER](#)
 - [optix_declarations.h](#), [63](#)
- [RT_INTERNAL_INVERSE_TRANSPOSE](#)
 - [optix_defines.h](#), [67](#)
- [RT_OBJECT_TO_WORLD](#)
 - [optix_defines.h](#), [67](#)
- [RT_OBJECTTYPE_BUFFER](#)
 - [optix_declarations.h](#), [63](#)
- [RT_OBJECTTYPE_FLOAT](#)
 - [optix_declarations.h](#), [64](#)
- [RT_OBJECTTYPE_FLOAT2](#)
 - [optix_declarations.h](#), [64](#)
- [RT_OBJECTTYPE_FLOAT3](#)
 - [optix_declarations.h](#), [64](#)
- [RT_OBJECTTYPE_FLOAT4](#)
 - [optix_declarations.h](#), [64](#)

- RT_OBJECTTYPE_GEOMETRY_GROUP
 - [optix_declarations.h, 63](#)
- RT_OBJECTTYPE_GEOMETRY_INSTANCE
 - [optix_declarations.h, 63](#)
- RT_OBJECTTYPE_GROUP
 - [optix_declarations.h, 63](#)
- RT_OBJECTTYPE_INT
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_INT2
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_INT3
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_INT4
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_MATRIX_FLOAT2x2
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_MATRIX_FLOAT2x3
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_MATRIX_FLOAT2x4
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_MATRIX_FLOAT3x2
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_MATRIX_FLOAT3x3
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_MATRIX_FLOAT3x4
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_MATRIX_FLOAT4x2
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_MATRIX_FLOAT4x3
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_MATRIX_FLOAT4x4
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_OBJECT
 - [optix_declarations.h, 63](#)
- RT_OBJECTTYPE_PROGRAM
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_SELECTOR
 - [optix_declarations.h, 63](#)
- RT_OBJECTTYPE_TEXTURE_SAMPLER
 - [optix_declarations.h, 63](#)
- RT_OBJECTTYPE_TRANSFORM
 - [optix_declarations.h, 63](#)
- RT_OBJECTTYPE_UNKNOWN
 - [optix_declarations.h, 63](#)
- RT_OBJECTTYPE_UNSIGNED_INT
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_UNSIGNED_INT2
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_UNSIGNED_INT3
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_UNSIGNED_INT4
 - [optix_declarations.h, 64](#)
- RT_OBJECTTYPE_USER
 - [optix_declarations.h, 64](#)
- RT_PROGRAM_ID_NULL
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_CLUSTER_URL
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_CONFIGURATIONNS
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_GPU_TOTAL_MEMORY
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_HEAD_NODE_URL
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_NAME
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_NUM_FREE_NODES
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_NUM_GPUS
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_NUM_RESERVED_NODES
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_NUM_TOTAL_NODES
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_ATTRIBUTE_STATUS
 - [optix_declarations.h, 64](#)
- RT_REMOTEDEVICE_STATUS_CONNECTED
 - [optix_declarations.h, 65](#)
- RT_REMOTEDEVICE_STATUS_DISCONNECTED
 - [optix_declarations.h, 65](#)
- RT_REMOTEDEVICE_STATUS_READY
 - [optix_declarations.h, 65](#)
- RT_REMOTEDEVICE_STATUS_RESERVED
 - [optix_declarations.h, 65](#)
- RT_SUCCESS
 - [optix_declarations.h, 65](#)
- RT_TARGET_GL_RENDER_BUFFER
 - [optix_declarations.h, 63](#)
- RT_TARGET_GL_TEXTURE_1D
 - [optix_declarations.h, 63](#)
- RT_TARGET_GL_TEXTURE_1D_ARRAY
 - [optix_declarations.h, 63](#)
- RT_TARGET_GL_TEXTURE_2D
 - [optix_declarations.h, 63](#)
- RT_TARGET_GL_TEXTURE_2D_ARRAY
 - [optix_declarations.h, 63](#)
- RT_TARGET_GL_TEXTURE_3D
 - [optix_declarations.h, 63](#)
- RT_TARGET_GL_TEXTURE_CUBE_MAP
 - [optix_declarations.h, 63](#)
- RT_TARGET_GL_TEXTURE_CUBE_MAP_ARRAY
 - [optix_declarations.h, 63](#)
- RT_TARGET_GL_TEXTURE_RECTANGLE
 - [optix_declarations.h, 63](#)
- RT_TEXTURE_ID_NULL
 - [optix_declarations.h, 66](#)
- RT_TEXTURE_INDEX_ARRAY_INDEX

- [optix_declarations.h](#), 66
- [RT_TEXTURE_INDEX_NORMALIZED_COORDINATES](#)
 - [optix_declarations.h](#), 66
- [RT_TEXTURE_READ_ELEMENT_TYPE](#)
 - [optix_declarations.h](#), 66
- [RT_TEXTURE_READ_ELEMENT_TYPE_SRGB](#)
 - [optix_declarations.h](#), 66
- [RT_TEXTURE_READ_NORMALIZED_FLOAT](#)
 - [optix_declarations.h](#), 66
- [RT_TEXTURE_READ_NORMALIZED_FLOAT_SRGB](#)
 - [optix_declarations.h](#), 66
- [RT_TIMEOUT_CALLBACK](#)
 - [optix_declarations.h](#), 65
- [RT_WORLD_TO_OBJECT](#)
 - [optix_defines.h](#), 67
- [RT_WRAP_CLAMP_TO_BORDER](#)
 - [optix_declarations.h](#), 66
- [RT_WRAP_CLAMP_TO_EDGE](#)
 - [optix_declarations.h](#), 66
- [RT_WRAP_MIRROR](#)
 - [optix_declarations.h](#), 66
- [RT_WRAP_REPEAT](#)
 - [optix_declarations.h](#), 66
- [RTP_BUFFER_FORMAT_HIT_BITMASK](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUFFER_FORMAT_HIT_T](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUFFER_FORMAT_HIT_T_TRIID](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID_U_V](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUFFER_FORMAT_HIT_T_TRIID_U_V](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUFFER_FORMAT_INDICES_INT3](#)
 - [optix_prime_declarations.h](#), 256
- [RTP_BUFFER_FORMAT_INDICES_INT3_MASK_INT](#)
 - [optix_prime_declarations.h](#), 256
- [RTP_BUFFER_FORMAT_INSTANCE_MODEL](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUFFER_FORMAT_RAY_ORIGIN_DIRECTION](#)
 - [optix_prime_declarations.h](#), 256
- [RTP_BUFFER_FORMAT_RAY_ORIGIN_MASK_DIRECTION_TMAX](#)
 - [optix_prime_declarations.h](#), 256
- [RTP_BUFFER_FORMAT_RAY_ORIGIN_TMIN_DIRECTION_TMAX](#)
 - [optix_prime_declarations.h](#), 256
- [RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x3](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x4](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUFFER_FORMAT_VERTEX_FLOAT3](#)
 - [optix_prime_declarations.h](#), 256
- [RTP_BUFFER_FORMAT_VERTEX_FLOAT4](#)
 - [optix_prime_declarations.h](#), 256
- [RTP_BUFFER_TYPE_CUDA_LINEAR](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUFFER_TYPE_HOST](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUILDER_PARAM_CHUNK_SIZE](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_BUILDER_PARAM_USE_CALLER_TRIANGLES](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_CONTEXT_TYPE_CPU](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_CONTEXT_TYPE_CUDA](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_ERROR_INVALID_CONTEXT](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_ERROR_INVALID_HANDLE](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_ERROR_INVALID_OPERATION](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_ERROR_INVALID_VALUE](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_ERROR_MEMORY_ALLOCATION_FAILED](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_ERROR_NOT_SUPPORTED](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_ERROR_OBJECT_CREATION_FAILED](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_ERROR_OUT_OF_MEMORY](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_ERROR_UNKNOWN](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_ERROR_VALIDATION_ERROR](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_MODEL_HINT_ASYNC](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_MODEL_HINT_MASK_UPDATE](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_MODEL_HINT_NONE](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_MODEL_HINT_USER_TRIANGLES_AFTER_COPY_SET](#)
 - [optix_prime_declarations.h](#), 257
- [RTP_QUERY_HINT_ASYNC](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_QUERY_HINT_NONE](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_QUERY_HINT_WATERTIGHT](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_QUERY_TYPE_ANY](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_QUERY_TYPE_CLOSEST](#)
 - [optix_prime_declarations.h](#), 258
- [RTP_SUCCESS](#)
 - [optix_prime_declarations.h](#), 258
- [RTU_INITOPTION_CPU_ONLY](#)
 - [optixu_traversal.h](#), 281
- [RTU_INITOPTION_CULL_BACKFACE](#)
 - [optixu_traversal.h](#), 281
- [RTU_INITOPTION_GPU_ONLY](#)

- optixu_traversal.h, [281](#)
- RTU_INITOPTION_NONE
 - optixu_traversal.h, [281](#)
- RTU_OPTION_INT_NUM_THREADS
 - optixu_traversal.h, [281](#)
- RTU_OUTPUT_BACKFACING
 - optixu_traversal.h, [281](#)
- RTU_OUTPUT_BARYCENTRIC
 - optixu_traversal.h, [281](#)
- RTU_OUTPUT_NONE
 - optixu_traversal.h, [281](#)
- RTU_OUTPUT_NORMAL
 - optixu_traversal.h, [281](#)
- RTU_QUERY_TYPE_ANY_HIT
 - optixu_traversal.h, [281](#)
- RTU_QUERY_TYPE_CLOSEST_HIT
 - optixu_traversal.h, [281](#)
- RTU_QUERY_TYPE_COUNT
 - optixu_traversal.h, [281](#)
- RTU_RAYFORMAT_COUNT
 - optixu_traversal.h, [282](#)
- RTU_RAYFORMAT_ORIGIN_DIRECTION_INTERLEAVED
 - optixu_traversal.h, [282](#)
- RTU_RAYFORMAT_ORIGIN_DIRECTION_TMIN_TMAX_INTERLEAVED
 - optixu_traversal.h, [282](#)
- RTU_TRIFORMAT_COUNT
 - optixu_traversal.h, [282](#)
- RTU_TRIFORMAT_MESH
 - optixu_traversal.h, [282](#)
- RTU_TRIFORMAT_TRIANGLE_SOUP
 - optixu_traversal.h, [282](#)
- RT_DEFAULT_MAX
 - optix_datatypes.h, [55](#)
- RT_PROGRAM
 - optix_device.h, [74](#)
- RTPbufferdesc
 - optix_prime.h, [242](#)
- RTPbufferformat
 - optix_prime_declarations.h, [256](#)
- RTPbuffertype
 - optix_prime_declarations.h, [257](#)
- RTPbuilderparam
 - optix_prime_declarations.h, [257](#)
- RTPcontext
 - optix_prime.h, [242](#)
- RTPcontexttype
 - optix_prime_declarations.h, [257](#)
- RTPmodel
 - optix_prime.h, [242](#)
- RTPmodelhint
 - optix_prime_declarations.h, [257](#)
- RTPquery
 - optix_prime.h, [242](#)
- RTPqueryhint
 - optix_prime_declarations.h, [257](#)
- RTPquerytype
 - optix_prime_declarations.h, [258](#)
- RTPresult
 - optix_prime_declarations.h, [258](#)
- RTU_CHECK_ERROR
 - optixu.h, [263](#)
- RTU_GROUP_ADD_CHILD
 - optixu.h, [263](#)
- RTUinitoptions
 - optixu_traversal.h, [281](#)
- RTUoption
 - optixu_traversal.h, [281](#)
- RTUoutput
 - optixu_traversal.h, [281](#)
- RTUquerytype
 - optixu_traversal.h, [281](#)
- RTUrayformat
 - optixu_traversal.h, [281](#)
- RTUtraversal
 - optixu_traversal.h, [280](#)
- RTUtraversalresult, [41](#)
 - prim_id, [41](#)
 - t, [41](#)
- RTUtriformat
 - optixu_traversal.h, [282](#)
- RTacceleration
 - optix_host.h, [102](#)
- RTbuffer
 - optix_host.h, [102](#)
- RTbufferattribute
 - optix_declarations.h, [60](#)
- RTbufferflag
 - optix_declarations.h, [60](#)
- RTbufferidnull
 - optix_declarations.h, [60](#)
- RTbuffermapflag
 - optix_declarations.h, [60](#)
- RTbuffertype
 - optix_declarations.h, [61](#)
- RTcontext
 - optix_host.h, [102](#)
- RTcontextattribute
 - optix_declarations.h, [61](#)
- RTdeviceattribute
 - optix_declarations.h, [61](#)
- RTexception
 - optix_declarations.h, [61](#)
- RTfiltermode
 - optix_declarations.h, [62](#)
- RTformat
 - optix_declarations.h, [62](#)
- RTgeometry
 - optix_host.h, [102](#)
- RTgeometrygroup
 - optix_host.h, [102](#)
- RTgeometryinstance
 - optix_host.h, [102](#)
- RTgltarget
 - optix_declarations.h, [63](#)

RTgroup
 optix_host.h, 102
RTmaterial
 optix_host.h, 102
RObject
 optix_host.h, 103
RObjecttype
 optix_declarations.h, 63
RTprogram
 optix_host.h, 103
RTprogramidnull
 optix_declarations.h, 64
RTremotedevice
 optix_host.h, 103
RTremotedeviceattribute
 optix_declarations.h, 64
RTremotedevicestatus
 optix_declarations.h, 64
RTresult
 optix_declarations.h, 65
RTselector
 optix_host.h, 103
RTtextureidnull
 optix_declarations.h, 65
RTtextureindexmode
 optix_declarations.h, 66
RTtexturereadmode
 optix_declarations.h, 66
RTtexturesampler
 optix_host.h, 103
RTtimeoutcallback
 optix_host.h, 103
RTtransform
 optix_host.h, 103
RTtransformflags
 optix_defines.h, 67
RTtransformkind
 optix_defines.h, 67
RTvariable
 optix_host.h, 103
RTwrapmode
 optix_declarations.h, 66
Ray, 38
 direction, 39
 origin, 39
 ray_type, 39
 tmax, 39
 tmin, 39
ray_type
 Ray, 39
registerGLTexture
 optix::TextureSamplerObj, 45
removeChild
 optix::GeometryGroupObj, 23
 optix::GroupObj, 28
 optix::SelectorObj, 43, 44
rotate
 optix::Matrix, 33
rtAccelerationCreate
 optix_host.h, 103
rtAccelerationDestroy
 optix_host.h, 104
rtAccelerationGetBuilder
 optix_host.h, 104
rtAccelerationGetContext
 optix_host.h, 105
rtAccelerationGetData
 optix_host.h, 105
rtAccelerationGetDataSize
 optix_host.h, 105
rtAccelerationGetProperty
 optix_host.h, 105
rtAccelerationGetTraverser
 optix_host.h, 106
rtAccelerationIsDirty
 optix_host.h, 106
rtAccelerationMarkDirty
 optix_host.h, 106
rtAccelerationSetBuilder
 optix_host.h, 107
rtAccelerationSetData
 optix_host.h, 108
rtAccelerationSetProperty
 optix_host.h, 108
rtAccelerationSetTraverser
 optix_host.h, 109
rtAccelerationValidate
 optix_host.h, 109
rtBuffer
 optix_device.h, 75
rtBufferBindProgressiveStream
 optix_host.h, 109
rtBufferCreate
 optix_host.h, 110
rtBufferCreateForCUDA
 optix_cuda_interop.h, 53
rtBufferCreateFromGLBO
 optix_gl_interop.h, 89
rtBufferDestroy
 optix_host.h, 111
rtBufferGLRegister
 optix_gl_interop.h, 90
rtBufferGLUnregister
 optix_gl_interop.h, 91
rtBufferGetAttribute
 optix_host.h, 111
rtBufferGetContext
 optix_host.h, 112
rtBufferGetDevicePointer
 optix_cuda_interop.h, 53
rtBufferGetDimensionality
 optix_host.h, 112
rtBufferGetElementSize
 optix_host.h, 113
rtBufferGetFormat
 optix_host.h, 113

rtBufferGetGLBOld
 optix_gl_interop.h, 90
rtBufferGetId
 optix_host.h, 114
rtBufferGetMipLevelCount
 optix_host.h, 114
rtBufferGetMipLevelSize1D
 optix_host.h, 115
rtBufferGetMipLevelSize2D
 optix_host.h, 115
rtBufferGetMipLevelSize3D
 optix_host.h, 116
rtBufferGetProgressiveUpdateReady
 optix_host.h, 116
rtBufferGetSize1D
 optix_host.h, 117
rtBufferGetSize2D
 optix_host.h, 117
rtBufferGetSize3D
 optix_host.h, 118
rtBufferGetSizev
 optix_host.h, 118
rtBufferId
 optix_device.h, 75
rtBufferMap
 optix_host.h, 119
rtBufferMapEx
 optix_host.h, 119
rtBufferMarkDirty
 optix_cuda_interop.h, 54
rtBufferSetAttribute
 optix_host.h, 120
rtBufferSetDevicePointer
 optix_cuda_interop.h, 54
rtBufferSetElementSize
 optix_host.h, 121
rtBufferSetFormat
 optix_host.h, 121
rtBufferSetMipLevelCount
 optix_host.h, 123
rtBufferSetSize1D
 optix_host.h, 123
rtBufferSetSize2D
 optix_host.h, 124
rtBufferSetSize3D
 optix_host.h, 124
rtBufferSetSizev
 optix_host.h, 125
rtBufferUnmap
 optix_host.h, 126
rtBufferUnmapEx
 optix_host.h, 126
rtBufferValidate
 optix_host.h, 127
rtCallableProgram
 optix_device.h, 75
rtCallableProgramId
 optix_device.h, 76
rtCallableProgramSizeofWrapper< T >, 40
rtCallableProgramSizeofWrapper< void >, 40
rtCallableProgramX
 optix_device.h, 76
rtContextCompile
 optix_host.h, 127
rtContextCreate
 optix_host.h, 127
rtContextDeclareVariable
 optix_host.h, 128
rtContextDestroy
 optix_host.h, 128
rtContextGetAttribute
 optix_host.h, 129
rtContextGetBufferFromId
 optix_host.h, 130
rtContextGetDeviceCount
 optix_host.h, 130
rtContextGetDevices
 optix_host.h, 131
rtContextGetEntryPointCount
 optix_host.h, 131
rtContextGetErrorString
 optix_host.h, 131
rtContextGetExceptionEnabled
 optix_host.h, 132
rtContextGetExceptionProgram
 optix_host.h, 132
rtContextGetMissProgram
 optix_host.h, 133
rtContextGetPrintBufferSize
 optix_host.h, 133
rtContextGetPrintEnabled
 optix_host.h, 134
rtContextGetPrintLaunchIndex
 optix_host.h, 134
rtContextGetProgramFromId
 optix_host.h, 135
rtContextGetRayGenerationProgram
 optix_host.h, 135
rtContextGetRayTypeCount
 optix_host.h, 136
rtContextGetRunningState
 optix_host.h, 136
rtContextGetStackSize
 optix_host.h, 136
rtContextGetTextureSamplerFromId
 optix_host.h, 137
rtContextGetVariable
 optix_host.h, 137
rtContextGetVariableCount
 optix_host.h, 138
rtContextLaunch1D
 optix_host.h, 138
rtContextLaunch2D
 optix_host.h, 139
rtContextLaunch3D
 optix_host.h, 139

rtContextLaunchProgressive2D
 optix_host.h, 139

rtContextQueryVariable
 optix_host.h, 140

rtContextRemoveVariable
 optix_host.h, 141

rtContextSetAttribute
 optix_host.h, 141

rtContextSetDevices
 optix_host.h, 142

rtContextSetEntryPointCount
 optix_host.h, 142

rtContextSetExceptionEnabled
 optix_host.h, 142

rtContextSetExceptionProgram
 optix_host.h, 144

rtContextSetMissProgram
 optix_host.h, 144

rtContextSetPrintBufferSize
 optix_host.h, 145

rtContextSetPrintEnabled
 optix_host.h, 145

rtContextSetPrintLaunchIndex
 optix_host.h, 145

rtContextSetRayGenerationProgram
 optix_host.h, 147

rtContextSetRayTypeCount
 optix_host.h, 147

rtContextSetRemoteDevice
 optix_host.h, 149

rtContextSetStackSize
 optix_host.h, 149

rtContextSetTimeoutCallback
 optix_host.h, 150

rtContextStopProgressive
 optix_host.h, 150

rtContextValidate
 optix_host.h, 151

rtDeclareAnnotation
 optix_device.h, 77

rtDeclareVariable
 optix_device.h, 78

rtDeviceGetAttribute
 optix_host.h, 151

rtDeviceGetDeviceCount
 optix_host.h, 152

rtDeviceGetWGLDevice
 optix_gl_interop.h, 91

rtGeometryCreate
 optix_host.h, 153

rtGeometryDeclareVariable
 optix_host.h, 153

rtGeometryDestroy
 optix_host.h, 154

rtGeometryGetBoundingBoxProgram
 optix_host.h, 154

rtGeometryGetContext
 optix_host.h, 155

rtGeometryGetIntersectionProgram
 optix_host.h, 155

rtGeometryGetPrimitiveCount
 optix_host.h, 156

rtGeometryGetPrimitiveIndexOffset
 optix_host.h, 156

rtGeometryGetVariable
 optix_host.h, 157

rtGeometryGetVariableCount
 optix_host.h, 157

rtGeometryGroupCreate
 optix_host.h, 158

rtGeometryGroupDestroy
 optix_host.h, 158

rtGeometryGroupGetAcceleration
 optix_host.h, 159

rtGeometryGroupGetChild
 optix_host.h, 159

rtGeometryGroupGetChildCount
 optix_host.h, 160

rtGeometryGroupGetContext
 optix_host.h, 160

rtGeometryGroupSetAcceleration
 optix_host.h, 161

rtGeometryGroupSetChild
 optix_host.h, 161

rtGeometryGroupSetChildCount
 optix_host.h, 163

rtGeometryGroupValidate
 optix_host.h, 163

rtGeometryInstanceCreate
 optix_host.h, 164

rtGeometryInstanceDeclareVariable
 optix_host.h, 164

rtGeometryInstanceDestroy
 optix_host.h, 166

rtGeometryInstanceGetContext
 optix_host.h, 166

rtGeometryInstanceGetGeometry
 optix_host.h, 167

rtGeometryInstanceGetMaterial
 optix_host.h, 167

rtGeometryInstanceGetMaterialCount
 optix_host.h, 168

rtGeometryInstanceGetVariable
 optix_host.h, 168

rtGeometryInstanceGetVariableCount
 optix_host.h, 169

rtGeometryInstanceQueryVariable
 optix_host.h, 169

rtGeometryInstanceRemoveVariable
 optix_host.h, 170

rtGeometryInstanceSetGeometry
 optix_host.h, 170

rtGeometryInstanceSetMaterial
 optix_host.h, 171

rtGeometryInstanceSetMaterialCount
 optix_host.h, 171

rtGeometryInstanceValidate
 optix_host.h, 172

rtGeometryIsDirty
 optix_host.h, 172

rtGeometryMarkDirty
 optix_host.h, 172

rtGeometryQueryVariable
 optix_host.h, 173

rtGeometryRemoveVariable
 optix_host.h, 173

rtGeometrySetBoundingBoxProgram
 optix_host.h, 174

rtGeometrySetIntersectionProgram
 optix_host.h, 174

rtGeometrySetPrimitiveCount
 optix_host.h, 175

rtGeometrySetPrimitiveIndexOffset
 optix_host.h, 175

rtGeometryValidate
 optix_host.h, 176

rtGetExceptionCode
 optix_device.h, 79

rtGetTransform
 optix_device.h, 79

rtGetVersion
 optix_host.h, 176

rtGroupCreate
 optix_host.h, 176

rtGroupDestroy
 optix_host.h, 178

rtGroupGetAcceleration
 optix_host.h, 178

rtGroupGetChild
 optix_host.h, 179

rtGroupGetChildCount
 optix_host.h, 179

rtGroupGetChildType
 optix_host.h, 179

rtGroupGetContext
 optix_host.h, 180

rtGroupSetAcceleration
 optix_host.h, 180

rtGroupSetChild
 optix_host.h, 181

rtGroupSetChildCount
 optix_host.h, 181

rtGroupValidate
 optix_host.h, 182

rtIgnoreIntersection
 optix_device.h, 80

rtIntersectChild
 optix_device.h, 80

rtMaterialCreate
 optix_host.h, 182

rtMaterialDeclareVariable
 optix_host.h, 183

rtMaterialDestroy
 optix_host.h, 183

rtMaterialGetAnyHitProgram
 optix_host.h, 184

rtMaterialGetClosestHitProgram
 optix_host.h, 184

rtMaterialGetContext
 optix_host.h, 185

rtMaterialGetVariable
 optix_host.h, 185

rtMaterialGetVariableCount
 optix_host.h, 186

rtMaterialQueryVariable
 optix_host.h, 186

rtMaterialRemoveVariable
 optix_host.h, 187

rtMaterialSetAnyHitProgram
 optix_host.h, 187

rtMaterialSetClosestHitProgram
 optix_host.h, 189

rtMaterialValidate
 optix_host.h, 189

rtObject, 40

rtPotentialIntersection
 optix_device.h, 81

rtPrintExceptionDetails
 optix_device.h, 81

rtPrintf
 optix_device.h, 81–85

rtProgramCreateFromPTXFile
 optix_host.h, 190

rtProgramCreateFromPTXString
 optix_host.h, 190

rtProgramDeclareVariable
 optix_host.h, 191

rtProgramDestroy
 optix_host.h, 192

rtProgramGetContext
 optix_host.h, 192

rtProgramGetId
 optix_host.h, 192

rtProgramGetVariable
 optix_host.h, 193

rtProgramGetVariableCount
 optix_host.h, 193

rtProgramQueryVariable
 optix_host.h, 194

rtProgramRemoveVariable
 optix_host.h, 194

rtProgramValidate
 optix_host.h, 195

rtRemoteDeviceCreate
 optix_host.h, 195

rtRemoteDeviceDestroy
 optix_host.h, 196

rtRemoteDeviceGetAttribute
 optix_host.h, 196

rtRemoteDeviceRelease
 optix_host.h, 198

rtRemoteDeviceReserve

- optix_host.h, 199
- rtReportIntersection
 - optix_device.h, 86
- rtSelectorCreate
 - optix_host.h, 199
- rtSelectorDeclareVariable
 - optix_host.h, 200
- rtSelectorDestroy
 - optix_host.h, 200
- rtSelectorGetChild
 - optix_host.h, 202
- rtSelectorGetChildCount
 - optix_host.h, 202
- rtSelectorGetChildType
 - optix_host.h, 203
- rtSelectorGetContext
 - optix_host.h, 203
- rtSelectorGetVariable
 - optix_host.h, 204
- rtSelectorGetVariableCount
 - optix_host.h, 204
- rtSelectorGetVisitProgram
 - optix_host.h, 205
- rtSelectorQueryVariable
 - optix_host.h, 205
- rtSelectorRemoveVariable
 - optix_host.h, 206
- rtSelectorSetChild
 - optix_host.h, 206
- rtSelectorSetChildCount
 - optix_host.h, 207
- rtSelectorSetVisitProgram
 - optix_host.h, 207
- rtSelectorValidate
 - optix_host.h, 208
- rtTerminateRay
 - optix_device.h, 86
- rtTextureSampler
 - optix_device.h, 79
- rtTextureSamplerCreate
 - optix_host.h, 208
- rtTextureSamplerCreateFromGLImage
 - optix_gl_interop.h, 92
- rtTextureSamplerDestroy
 - optix_host.h, 209
- rtTextureSamplerGLRegister
 - optix_gl_interop.h, 93
- rtTextureSamplerGLUnregister
 - optix_gl_interop.h, 94
- rtTextureSamplerGetArraySize
 - optix_host.h, 209
- rtTextureSamplerGetBuffer
 - optix_host.h, 209
- rtTextureSamplerGetContext
 - optix_host.h, 210
- rtTextureSamplerGetFilteringModes
 - optix_host.h, 210
- rtTextureSamplerGetGLImageId
 - optix_gl_interop.h, 93
- rtTextureSamplerGetId
 - optix_host.h, 211
- rtTextureSamplerGetIndexingMode
 - optix_host.h, 211
- rtTextureSamplerGetMaxAnisotropy
 - optix_host.h, 212
- rtTextureSamplerGetMipLevelBias
 - optix_host.h, 212
- rtTextureSamplerGetMipLevelClamp
 - optix_host.h, 213
- rtTextureSamplerGetMipLevelCount
 - optix_host.h, 213
- rtTextureSamplerGetReadMode
 - optix_host.h, 213
- rtTextureSamplerGetWrapMode
 - optix_host.h, 214
- rtTextureSamplerSetArraySize
 - optix_host.h, 214
- rtTextureSamplerSetBuffer
 - optix_host.h, 214
- rtTextureSamplerSetFilteringModes
 - optix_host.h, 215
- rtTextureSamplerSetIndexingMode
 - optix_host.h, 215
- rtTextureSamplerSetMaxAnisotropy
 - optix_host.h, 216
- rtTextureSamplerSetMipLevelBias
 - optix_host.h, 216
- rtTextureSamplerSetMipLevelClamp
 - optix_host.h, 217
- rtTextureSamplerSetMipLevelCount
 - optix_host.h, 217
- rtTextureSamplerSetReadMode
 - optix_host.h, 217
- rtTextureSamplerSetWrapMode
 - optix_host.h, 218
- rtTextureSamplerValidate
 - optix_host.h, 219
- rtThrow
 - optix_device.h, 86
- rtTrace
 - optix_device.h, 87
- rtTransformCreate
 - optix_host.h, 219
- rtTransformDestroy
 - optix_host.h, 220
- rtTransformGetChild
 - optix_host.h, 220
- rtTransformGetChildType
 - optix_host.h, 221
- rtTransformGetContext
 - optix_host.h, 221
- rtTransformGetMatrix
 - optix_host.h, 222
- rtTransformNormal
 - optix_device.h, 87
- rtTransformPoint

optix_device.h, [87](#)
rtTransformSetChild
 optix_host.h, [222](#)
rtTransformSetMatrix
 optix_host.h, [223](#)
rtTransformValidate
 optix_host.h, [224](#)
rtTransformVector
 optix_device.h, [88](#)
rtVariableGet1f
 optix_host.h, [224](#)
rtVariableGet1fv
 optix_host.h, [225](#)
rtVariableGet1i
 optix_host.h, [225](#)
rtVariableGet1iv
 optix_host.h, [225](#)
rtVariableGet1ui
 optix_host.h, [225](#)
rtVariableGet1uiv
 optix_host.h, [225](#)
rtVariableGet2f
 optix_host.h, [225](#)
rtVariableGet2fv
 optix_host.h, [226](#)
rtVariableGet2i
 optix_host.h, [226](#)
rtVariableGet2iv
 optix_host.h, [226](#)
rtVariableGet2ui
 optix_host.h, [226](#)
rtVariableGet2uiv
 optix_host.h, [226](#)
rtVariableGet3f
 optix_host.h, [226](#)
rtVariableGet3fv
 optix_host.h, [226](#)
rtVariableGet3i
 optix_host.h, [227](#)
rtVariableGet3iv
 optix_host.h, [227](#)
rtVariableGet3ui
 optix_host.h, [227](#)
rtVariableGet3uiv
 optix_host.h, [227](#)
rtVariableGet4f
 optix_host.h, [227](#)
rtVariableGet4fv
 optix_host.h, [227](#)
rtVariableGet4i
 optix_host.h, [228](#)
rtVariableGet4iv
 optix_host.h, [228](#)
rtVariableGet4ui
 optix_host.h, [228](#)
rtVariableGet4uiv
 optix_host.h, [228](#)
rtVariableGetAnnotation
 optix_host.h, [228](#)
rtVariableGetContext
 optix_host.h, [229](#)
rtVariableGetMatrix2x2fv
 optix_host.h, [229](#)
rtVariableGetMatrix2x3fv
 optix_host.h, [229](#)
rtVariableGetMatrix2x4fv
 optix_host.h, [229](#)
rtVariableGetMatrix3x2fv
 optix_host.h, [230](#)
rtVariableGetMatrix3x3fv
 optix_host.h, [230](#)
rtVariableGetMatrix3x4fv
 optix_host.h, [230](#)
rtVariableGetMatrix4x2fv
 optix_host.h, [230](#)
rtVariableGetMatrix4x3fv
 optix_host.h, [230](#)
rtVariableGetMatrix4x4fv
 optix_host.h, [230](#)
rtVariableGetName
 optix_host.h, [230](#)
rtVariableGetObject
 optix_host.h, [231](#)
rtVariableGetSize
 optix_host.h, [231](#)
rtVariableGetType
 optix_host.h, [232](#)
rtVariableGetUserData
 optix_host.h, [233](#)
rtVariableSet1f
 optix_host.h, [234](#)
rtVariableSet1fv
 optix_host.h, [234](#)
rtVariableSet1i
 optix_host.h, [235](#)
rtVariableSet1iv
 optix_host.h, [235](#)
rtVariableSet1ui
 optix_host.h, [235](#)
rtVariableSet1uiv
 optix_host.h, [235](#)
rtVariableSet2f
 optix_host.h, [235](#)
rtVariableSet2fv
 optix_host.h, [235](#)
rtVariableSet2i
 optix_host.h, [235](#)
rtVariableSet2iv
 optix_host.h, [236](#)
rtVariableSet2ui
 optix_host.h, [236](#)
rtVariableSet2uiv
 optix_host.h, [236](#)
rtVariableSet3f
 optix_host.h, [236](#)
rtVariableSet3fv

optix_host.h, [236](#)
 rtVariableSet3i
 optix_host.h, [236](#)
 rtVariableSet3iv
 optix_host.h, [236](#)
 rtVariableSet3ui
 optix_host.h, [237](#)
 rtVariableSet3uiv
 optix_host.h, [237](#)
 rtVariableSet4f
 optix_host.h, [237](#)
 rtVariableSet4fv
 optix_host.h, [237](#)
 rtVariableSet4i
 optix_host.h, [237](#)
 rtVariableSet4iv
 optix_host.h, [237](#)
 rtVariableSet4ui
 optix_host.h, [237](#)
 rtVariableSet4uiv
 optix_host.h, [238](#)
 rtVariableSetMatrix2x2fv
 optix_host.h, [238](#)
 rtVariableSetMatrix2x3fv
 optix_host.h, [238](#)
 rtVariableSetMatrix2x4fv
 optix_host.h, [238](#)
 rtVariableSetMatrix3x2fv
 optix_host.h, [238](#)
 rtVariableSetMatrix3x3fv
 optix_host.h, [238](#)
 rtVariableSetMatrix3x4fv
 optix_host.h, [239](#)
 rtVariableSetMatrix4x2fv
 optix_host.h, [239](#)
 rtVariableSetMatrix4x3fv
 optix_host.h, [239](#)
 rtVariableSetMatrix4x4fv
 optix_host.h, [239](#)
 rtVariableSetObject
 optix_host.h, [239](#)
 rtVariableSetUserData
 optix_host.h, [240](#)
 rti_internal_callableprogram::CPArgVoid, [20](#)
 rti_internal_callableprogram::check_is_CPArgVoid<
 Condition, Dummy >, [13](#)
 rti_internal_callableprogram::check_is_CPArgVoid<
 false, IntentionalError >, [14](#)
 rti_internal_callableprogram::is_CPArgVoid< CPArg-
 Void >, [29](#)
 rti_internal_callableprogram::is_CPArgVoid< T1 >, [29](#)
 rti_internal_typeinfo::rti_typeenum< optix::bound-
 CallableProgramId< T > >, [40](#)
 rti_internal_typeinfo::rti_typeenum< optix::callable-
 ProgramId< T > >, [40](#)
 rti_internal_typeinfo::rti_typeenum< T >, [40](#)
 rti_internal_typeinfo::rti_typeinfo, [40](#)
 rtpBufferDescCreate
 optix_prime.h, [242](#)
 rtpBufferDescDestroy
 optix_prime.h, [242](#)
 rtpBufferDescGetContext
 optix_prime.h, [243](#)
 rtpBufferDescSetCudaDeviceNumber
 optix_prime.h, [243](#)
 rtpBufferDescSetRange
 optix_prime.h, [243](#)
 rtpBufferDescSetStride
 optix_prime.h, [244](#)
 rtpContextCreate
 optix_prime.h, [244](#)
 rtpContextDestroy
 optix_prime.h, [245](#)
 rtpContextGetLastErrorString
 optix_prime.h, [245](#)
 rtpContextSetCpuThreads
 optix_prime.h, [245](#)
 rtpContextSetCudaDeviceNumbers
 optix_prime.h, [246](#)
 rtpGetErrorString
 optix_prime.h, [246](#)
 rtpGetVersion
 optix_prime.h, [247](#)
 rtpGetVersionString
 optix_prime.h, [247](#)
 rtpHostBufferLock
 optix_prime.h, [247](#)
 rtpHostBufferUnlock
 optix_prime.h, [247](#)
 rtpModelCopy
 optix_prime.h, [248](#)
 rtpModelCreate
 optix_prime.h, [248](#)
 rtpModelDestroy
 optix_prime.h, [248](#)
 rtpModelFinish
 optix_prime.h, [249](#)
 rtpModelGetContext
 optix_prime.h, [249](#)
 rtpModelGetFinished
 optix_prime.h, [249](#)
 rtpModelSetBuilderParameter
 optix_prime.h, [250](#)
 rtpModelSetInstances
 optix_prime.h, [250](#)
 rtpModelSetTriangles
 optix_prime.h, [251](#)
 rtpModelUpdate
 optix_prime.h, [251](#)
 rtpQueryCreate
 optix_prime.h, [252](#)
 rtpQueryDestroy
 optix_prime.h, [252](#)
 rtpQueryExecute
 optix_prime.h, [253](#)
 rtpQueryFinish

- optix_prime.h, 253
- rtpQueryGetContext
 - optix_prime.h, 253
- rtpQueryGetFinished
 - optix_prime.h, 254
- rtpQuerySetCudaStream
 - optix_prime.h, 254
- rtpQuerySetHits
 - optix_prime.h, 254
- rtpQuerySetRays
 - optix_prime.h, 255
- rtuCUDACompileFile
 - optixu.h, 265
- rtuCUDACompileString
 - optixu.h, 265
- rtuCUDAGetCompileResult
 - optixu.h, 265
- rtuCreateClusteredMesh
 - optixu.h, 263
- rtuCreateClusteredMeshExt
 - optixu.h, 264
- rtuGeometryGroupAddChild
 - optixu.h, 266
- rtuGeometryGroupGetChildIndex
 - optixu.h, 266
- rtuGeometryGroupRemoveChild
 - optixu.h, 266
- rtuGeometryGroupRemoveChildByIndex
 - optixu.h, 266
- rtuGetSizeForRTformat
 - optixu.h, 266
- rtuGroupAddChild
 - optixu.h, 266
- rtuGroupGetChildIndex
 - optixu.h, 266
- rtuGroupRemoveChild
 - optixu.h, 266
- rtuGroupRemoveChildByIndex
 - optixu.h, 266
- rtuNameForType
 - optixu.h, 266
- rtuSelectorAddChild
 - optixu.h, 268
- rtuSelectorGetChildIndex
 - optixu.h, 268
- rtuSelectorRemoveChild
 - optixu.h, 268
- rtuSelectorRemoveChildByIndex
 - optixu.h, 268
- rtuTransformGetChild
 - optixu.h, 268
- rtuTransformGetChildType
 - optixu.h, 268
- rtuTransformSetChild
 - optixu.h, 268
- rtuTraversalCreate
 - optixu_traversal.h, 282
- rtuTraversalDestroy
 - optixu_traversal.h, 282
- rtuTraversalGetAccelData
 - optixu_traversal.h, 282
- rtuTraversalGetAccelDataSize
 - optixu_traversal.h, 283
- rtuTraversalGetErrorString
 - optixu_traversal.h, 283
- rtuTraversalMapOutput
 - optixu_traversal.h, 283
- rtuTraversalMapRays
 - optixu_traversal.h, 283
- rtuTraversalMapResults
 - optixu_traversal.h, 283
- rtuTraversalPreprocess
 - optixu_traversal.h, 284
- rtuTraversalSetAccelData
 - optixu_traversal.h, 284
- rtuTraversalSetMesh
 - optixu_traversal.h, 284
- rtuTraversalSetOption
 - optixu_traversal.h, 284
- rtuTraversalSetTriangles
 - optixu_traversal.h, 284
- rtuTraversalTraverse
 - optixu_traversal.h, 285
- rtuTraversalUnmapOutput
 - optixu_traversal.h, 285
- rtuTraversalUnmapRays
 - optixu_traversal.h, 285
- rtuTraversalUnmapResults
 - optixu_traversal.h, 285
- scale
 - optix::Matrix, 33
- set
 - optix::Aabb, 5
- setAcceleration
 - optix::GeometryGroupObj, 24
 - optix::GroupObj, 28
- setAttribute
 - optix::BufferObj, 13
- setBoundingBoxProgram
 - optix::GeometryObj, 26
- setBuffer
 - optix::TextureSamplerObj, 45
- setBuilderParameter
 - optix::prime::ModelObj, 35
- setCPUNumThreads
 - optix::ContextObj, 20
- setChild
 - optix::TransformObj, 46
- setChildCount
 - optix::GeometryGroupObj, 24
 - optix::GroupObj, 28
 - optix::SelectorObj, 44
- setClosestHitProgram
 - optix::MaterialObj, 31
- setCol
 - optix::Matrix, 33

- setDevices
 - optix::ContextObj, [20](#)
- setFormat
 - optix::BufferObj, [13](#)
- setGeometry
 - optix::GeometryInstanceObj, [25](#)
- setInstances
 - optix::prime::ModelObj, [35](#)
- setMatrix
 - optix::TransformObj, [46](#)
- setMipLevelCount
 - optix::TextureSamplerObj, [45](#)
- setPrimitiveCount
 - optix::GeometryObj, [27](#)
- setPrimitiveIndexOffset
 - optix::GeometryObj, [27](#)
- setPrintEnabled
 - optix::ContextObj, [20](#)
- setProperty
 - optix::AccelerationObj, [7](#)
- setRayGenerationProgram
 - optix::ContextObj, [20](#)
- setRow
 - optix::Matrix, [33](#)
- setSize
 - optix::BufferObj, [13](#)
- setStackSize
 - optix::ContextObj, [20](#)
- setTimeoutCallback
 - optix::ContextObj, [20](#)
- setTriangles
 - optix::prime::ModelObj, [35](#)
- setVisitProgram
 - optix::SelectorObj, [44](#)
- signedDistance
 - optix::Aabb, [5](#)
- t
 - RTUtraversalresult, [41](#)
- take
 - optix::Handle, [29](#)
- tmax
 - Ray, [39](#)
- tmin
 - Ray, [39](#)
- translate
 - optix::Matrix, [34](#)
- transpose
 - optix::Matrix, [34](#)
- valid
 - optix::Aabb, [5](#)
- volume
 - optix::Aabb, [6](#)