



NVIDIA OptiX

API Reference Manual

27 February 2019

Version 6.0

Contents

1 OptiX Components	1
2 Module Index	1
2.1 Modules	1
3 Hierarchical Index	2
3.1 Class Hierarchy	2
4 Class Index	4
4.1 Class List	4
5 Module Documentation	8
5.1 OptiX API Reference	8
5.2 Context handling functions	9
5.3 rtContextLaunch functions	41
5.4 GeometryGroup handling functions	43
5.5 GroupNode functions	53
5.6 SelectorNode functions	62
5.7 TransformNode functions	73
5.8 Acceleration functions	86
5.9 GeometryInstance functions	94
5.10 Geometry functions	106
5.11 GeometryTriangles functions	123
5.12 Material functions	148
5.13 Program functions	158
5.14 Buffer functions	169
5.15 TextureSampler functions	200
5.16 Variable functions	218
5.17 Variable setters	225
5.18 Variable getters	240
5.19 CommandList functions	255
5.20 Context-free functions	265
5.21 CUDA C Reference	270
5.22 OptiX CUDA C declarations	271
5.23 OptiX basic types	278

5.24 OptiX CUDA C functions	280
5.25 Texture fetch functions	290
5.26 rtPrintf functions	291
5.27 OptiXpp wrapper	301
5.28 rtu API	304
5.29 rtu Traversal API	310
5.30 OptiX Prime API Reference	311
5.31 Context	312
5.32 Buffer descriptor	316
5.33 Model	320
5.34 Query	327
5.35 Miscellaneous functions	332
5.36 OptiX Prime++ wrapper	335
5.37 OptiX Interoperability Types	336
5.38 OpenGL Texture Formats	337
5.39 DXGI Texture Formats	338
6 Namespace Documentation	339
6.1 optix Namespace Reference	339
6.2 optix::detail Namespace Reference	543
6.3 optix::prime Namespace Reference	543
6.4 optixu Namespace Reference	544
6.5 rti_internal_callableprogram Namespace Reference	544
6.6 rti_internal_typeinfo Namespace Reference	544
7 Class Documentation	545
7.1 optix::Aabb Class Reference	545
7.2 optix::AccelerationObj Class Reference	550
7.3 optix::APIObj Class Reference	553
7.4 optix::boundCallableProgramId< T > Class Template Reference	555
7.5 optix::buffer< T, Dim > Struct Template Reference	555
7.6 optix::prime::BufferDescObj Class Reference	557
7.7 optix::bufferId< T, Dim > Struct Template Reference	558
7.8 optix::BufferObj Class Reference	562
7.9 optix::callableProgramId< T > Class Template Reference	568
7.10 rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T > Struct Template Reference	569

7.11 rti_internal_callableprogram::check_is_CPAVoid< Condition, Dummy > Struct Template Reference	578
7.12 rti_internal_callableprogram::check_is_CPAVoid< false, IntentionalError > Struct Template Reference	578
7.13 optix::CommandListObj Class Reference	579
7.14 optix::prime::ContextObj Class Reference	581
7.15 optix::ContextObj Class Reference	583
7.16 rti_internal_callableprogram::CPArgVoid Class Reference	601
7.17 optix::DestroyableObj Class Reference	601
7.18 optix::prime::Exception Class Reference	604
7.19 optix::Exception Class Reference	605
7.20 optix::GeometryGroupObj Class Reference	606
7.21 optix::GeometryInstanceObj Class Reference	610
7.22 optix::GeometryObj Class Reference	614
7.23 optix::GeometryTrianglesObj Class Reference	619
7.24 optix::GroupObj Class Reference	626
7.25 optix::Handle< T > Class Template Reference	630
7.26 rti_internal_callableprogram::is_CPAVoid< T1 > Struct Template Reference	633
7.27 rti_internal_callableprogram::is_CPAVoid< CPArgVoid > Struct Template Reference	633
7.28 optix::markedCallableProgramId< T > Class Template Reference	633
7.29 rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T > Struct Template Reference	633
7.30 optix::MaterialObj Class Reference	642
7.31 optix::Matrix< M, N > Class Template Reference	646
7.32 optix::prime::ModelObj Class Reference	651
7.33 optix::Onb Struct Reference	654
7.34 OptiXAPI Struct Reference	655
7.35 optix::PostprocessingStageObj Class Reference	667
7.36 optix::ProgramObj Class Reference	670
7.37 optix::PtxEncryption Class Reference	672
7.38 optix::Quaternion Class Reference	674
7.39 optix::prime::QueryObj Class Reference	675
7.40 Ray Struct Reference	677
7.41 rtCallableProgramSizeofWrapper< T > Struct Template Reference	679
7.42 rtCallableProgramSizeofWrapper< void > Struct Template Reference	679
7.43 rti_internal_typeinfo::rti_typeenum< T > Struct Template Reference	679
7.44 rti_internal_typeinfo::rti_typeenum< optix::boundCallableProgramId< T > > Struct Template Reference	680
7.45 rti_internal_typeinfo::rti_typeenum< optix::callableProgramId< T > > Struct Template Reference	680

7.46 rti_internal_typeinfo::rti_typeinfo Struct Reference	680
7.47 rtObject Struct Reference	681
7.48 optix::ScopedObj Class Reference	681
7.49 optix::SelectorObj Class Reference	685
7.50 optix::TextureSamplerObj Class Reference	688
7.51 optix::TransformObj Class Reference	694
7.52 optix::buffer< T, Dim >::type< T2 > Struct Template Reference	697
7.53 optix::VariableObj Class Reference	697
7.54 optix::VectorDim< DIM > Struct Template Reference	713
7.55 optix::VectorDim< 2 > Struct Template Reference	713
7.56 optix::VectorDim< 3 > Struct Template Reference	713
7.57 optix::VectorDim< 4 > Struct Template Reference	713
7.58 optix::VectorTypes< T, Dim > Struct Template Reference	714
7.59 optix::VectorTypes< float, 1 > Struct Template Reference	714
7.60 optix::VectorTypes< float, 2 > Struct Template Reference	714
7.61 optix::VectorTypes< float, 3 > Struct Template Reference	715
7.62 optix::VectorTypes< float, 4 > Struct Template Reference	715
7.63 optix::VectorTypes< int, 1 > Struct Template Reference	716
7.64 optix::VectorTypes< int, 2 > Struct Template Reference	716
7.65 optix::VectorTypes< int, 3 > Struct Template Reference	717
7.66 optix::VectorTypes< int, 4 > Struct Template Reference	717
7.67 optix::VectorTypes< unsigned int, 1 > Struct Template Reference	718
7.68 optix::VectorTypes< unsigned int, 2 > Struct Template Reference	718
7.69 optix::VectorTypes< unsigned int, 3 > Struct Template Reference	719
7.70 optix::VectorTypes< unsigned int, 4 > Struct Template Reference	719
8 File Documentation	720
8.1 Atom.h File Reference	720
8.2 doxygen_hierarchy.h File Reference	721
8.3 footer.tex File Reference	721
8.4 Handle.h File Reference	721
8.5 header.tex File Reference	721
8.6 interop_types.h File Reference	722
8.7 nvoptix.h File Reference	722
8.8 optix.h File Reference	742

8.9 optix_cuda.h File Reference	742
8.10 optix_cuda_interop.h File Reference	742
8.11 optix_datatypes.h File Reference	743
8.12 optix_declarations.h File Reference	744
8.13 optix_declarations_private.h File Reference	760
8.14 optixDefines.h File Reference	762
8.15 optix_device.h File Reference	764
8.16 optix_gl_interop.h File Reference	776
8.17 optix_host.h File Reference	777
8.18 optix_internal.h File Reference	795
8.19 optix_math.h File Reference	797
8.20 optix_prime.h File Reference	797
8.21 optix_prime_declarations.h File Reference	799
8.22 optix_primepp.h File Reference	803
8.23 optix_ptx_encryption.h File Reference	804
8.24 optix_sizet.h File Reference	805
8.25 optix_world.h File Reference	806
8.26 optixpp.h File Reference	806
8.27 optixpp_namespace.h File Reference	807
8.28 optixu.h File Reference	811
8.29 optixu_aabb.h File Reference	813
8.30 optixu_aabb_namespace.h File Reference	813
8.31 optixu_math.h File Reference	813
8.32 optixu_math_namespace.h File Reference	814
8.33 optixu_math_stream.h File Reference	831
8.34 optixu_math_stream_namespace.h File Reference	831
8.35 optixu_matrix.h File Reference	832
8.36 optixu_matrix_namespace.h File Reference	832
8.37 optixu_quaternion.h File Reference	835
8.38 optixu_quaternion_namespace.h File Reference	835
8.39 optixu_vector_functions.h File Reference	836
8.40 optixu_vector_types.h File Reference	836
8.41 Ref.h File Reference	836
8.42 refman.tex File Reference	836

1 OptiX Components

An extensive description of OptiX framework components and their features can be found in the document *OptiX_Programming_Guide.pdf* shipped with the SDK.

Components API Reference

OptiX - a scalable framework for building ray tracing applications.

See [OptiX API Reference](#) for details .

OptiXpp - C++ wrapper around OptiX objects and handling functions.

See [OptiXpp wrapper](#) for details .

OptiXu - simple API for performing raytracing queries using OptiX or the CPU. Also includes the rtuTraversal API subset for ray/triangle intersection.

See [CUDA C Reference](#) and [rtu API](#) for details .

OptiX Prime - high performance API for intersecting a set of rays against a set of triangles.

See [OptiX Prime API Reference](#) for details .

OptiX Prime++ - C++ wrapper around OptiX Prime objects and handling functions.

See [OptiX Prime++ wrapper](#) for details .

2 Module Index

2.1 Modules

Here is a list of all modules:

OptiX API Reference	8
Context handling functions	9
rtContextLaunch functions	41
GeometryGroup handling functions	43
GroupNode functions	53
SelectorNode functions	62
TransformNode functions	73
Acceleration functions	86
GeometryInstance functions	94
Geometry functions	106
GeometryTriangles functions	123
Material functions	148

Program functions	158
Buffer functions	169
TextureSampler functions	200
Variable functions	218
Variable setters	225
Variable getters	240
CommandList functions	255
Context-free functions	265
CUDA C Reference	270
OptiX CUDA C declarations	271
OptiX basic types	278
OptiX CUDA C functions	280
Texture fetch functions	290
rtPrintf functions	291
OptiXpp wrapper	301
rtu API	304
rtu Traversal API	310
OptiX Prime API Reference	311
Context	312
Buffer descriptor	316
Model	320
Query	327
Miscellaneous functions	332
OptiX Prime++ wrapper	335
OptiX Interoperability Types	336
OpenGL Texture Formats	337
DXGI Texture Formats	338

3 Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

optix::Aabb	545
optix::APIObj	553
optix::DestroyableObj	601
optix::AccelerationObj	550
optix::BufferObj	562

optix::CommandListObj	579
optix::GeometryGroupObj	606
optix::GeometryTrianglesObj	619
optix::GroupObj	626
optix::PostprocessingStageObj	667
optix::ScopedObj	681
optix::ContextObj	583
optix::GeometryInstanceObj	610
optix::GeometryObj	614
optix::MaterialObj	642
optix::ProgramObj	670
optix::SelectorObj	685
optix::TextureSamplerObj	688
optix::TransformObj	694
optix::VariableObj	697
optix::boundCallableProgramId< T >	555
optix::buffer< T, Dim >	555
optix::bufferId< T, Dim >	558
optix::callableProgramId< T >	568
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >	568
rti_internal_callableprogram::check_is_CPAVoid< Condition, Dummy >	578
rti_internal_callableprogram::check_is_CPAVoid< false, IntentionalError >	578
rti_internal_callableprogram::CPAArgVoid	601
std::exception[external]	
optix::Exception	605
optix::prime::Exception	604
optix::Handle< T >	630
optix::Handle< ContextObj >	630
optix::Handle< ModelObj >	630
rti_internal_callableprogram::is_CPAVoid< T1 >	633
rti_internal_callableprogram::is_CPAVoid< CPAArgVoid >	633
optix::markedCallableProgramId< T >	633
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >	634
optix::Matrix< M, N >	646
optix::Onb	654
OptiXAPI	655
optix::PtxEncryption	672

optix::Quaternion	674
Ray	677
RefCountedObj	
optix::prime::BufferDescObj	557
optix::prime::ContextObj	581
optix::prime::ModelObj	651
optix::prime::QueryObj	675
rtCallableProgramSizeofWrapper< T >	679
rtCallableProgramSizeofWrapper< void >	679
rti_internal_typeinfo::rti_typeenum< T >	679
rti_internal_typeinfo::rti_typeenum< optix::boundCallableProgramId< T > >	680
rti_internal_typeinfo::rti_typeenum< optix::callableProgramId< T > >	680
rti_internal_typeinfo::rti_typeinfo	680
rtObject	681
optix::buffer< T, Dim >::type< T2 >	697
optix::VectorDim< DIM >	713
optix::VectorDim< 2 >	713
optix::VectorDim< 3 >	713
optix::VectorDim< 4 >	713
optix::VectorTypes< T, Dim >	714
optix::VectorTypes< float, 1 >	714
optix::VectorTypes< float, 2 >	714
optix::VectorTypes< float, 3 >	715
optix::VectorTypes< float, 4 >	715
optix::VectorTypes< int, 1 >	716
optix::VectorTypes< int, 2 >	716
optix::VectorTypes< int, 3 >	717
optix::VectorTypes< int, 4 >	717
optix::VectorTypes< unsigned int, 1 >	718
optix::VectorTypes< unsigned int, 2 >	718
optix::VectorTypes< unsigned int, 3 >	719
optix::VectorTypes< unsigned int, 4 >	719

4 Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

optix::Aabb	Axis-aligned bounding box	545
optix::AccelerationObj	Acceleration wraps the OptiX C API RTacceleration opaque type and its associated function set	550
optix::APIObj	Base class for all reference counted wrappers around OptiX C API opaque types	553
optix::boundCallableProgramId< T >		555
optix::buffer< T, Dim >		555
optix::prime::BufferDescObj	Encapsulates an OptiX Prime buffer descriptor	557
optix::bufferId< T, Dim >	BufferId is a host version of the device side bufferId	558
optix::BufferObj	Buffer wraps the OptiX C API RTbuffer opaque type and its associated function set	562
optix::callableProgramId< T >		568
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T >		568
rti_internal_callableprogram::check_is_CPAVoid< Condition, Dummy >		578
rti_internal_callableprogram::check_is_CPAVoid< false, IntentionalError >		578
optix::CommandListObj	CommandList wraps the OptiX C API RTcommandlist opaque type and its associated function set	579
optix::prime::ContextObj	Wraps the OptiX Prime C API RTPcontext opaque type and its associated function set representing an OptiX Prime context	581
optix::ContextObj	Context object wraps the OptiX C API RTcontext opaque type and its associated function set	583
rti_internal_callableprogram::CPArgVoid		601
optix::DestroyableObj	Base class for all wrapper objects which can be destroyed and validated	601
optix::prime::Exception	Encapsulates an OptiX Prime exception	604
optix::Exception	Exception class for error reporting from the OptiXpp API	605
optix::GeometryGroupObj	GeometryGroup wraps the OptiX C API RTgeometrygroup opaque type and its associated function set	606
optix::GeometryInstanceObj	GeometryInstance wraps the OptiX C API RTgeometryinstance acceleration opaque type and its associated function set	610

optix::GeometryObj	Geometry wraps the OptiX C API RTgeometry opaque type and its associated function set	614
optix::GeometryTrianglesObj	GeometryTriangles wraps the OptiX C API RTgeometrytriangles opaque type and its associated function set	619
optix::GroupObj	Group wraps the OptiX C API RTgroup opaque type and its associated function set	626
optix::Handle< T >	The Handle class is a reference counted handle class used to manipulate API objects	630
rti_internal_callableprogram::is_CPAVoid< T1 >		633
rti_internal_callableprogram::is_CPAVoid< CPAVoid >		633
optix::markedCallableProgramId< T >		633
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T >		634
optix::MaterialObj	Material wraps the OptiX C API RTmaterial opaque type and its associated function set	642
optix::Matrix< M, N >	A matrix with M rows and N columns	646
optix::prime::ModelObj	Encapsulates an OptiX Prime model	651
optix::Onb	Orthonormal basis	654
OptiXAPI		655
optix::PostprocessingStageObj	PostProcessingStage wraps the OptiX C API RTpostprocessingstage opaque type and its associated function set	667
optix::ProgramObj	Program object wraps the OptiX C API RTprogram opaque type and its associated function set	670
optix::PtxEncryption		672
optix::Quaternion	Quaternion	674
optix::prime::QueryObj	Encapsulates an OptiX Prime query	675
Ray		
 Ray class		677
 rtCallableProgramSizeofWrapper< T >		679
 rtCallableProgramSizeofWrapper< void >		679
 rti_internal_typeinfo::rti_typeenum< T >		679
 rti_internal_typeinfo::rti_typeenum< optix::boundCallableProgramId< T > >		680

rti_internal_typeinfo::rti_typeenum< optix::callableProgramId< T > >	680
rti_internal_typeinfo::rti_typeinfo	680
rtObject	
Opaque handle to a OptiX object	681
optix::ScopedObj	
Base class for all objects which are OptiX variable containers	681
optix::SelectorObj	
Selector wraps the OptiX C API RTselector opaque type and its associated function set	685
optix::TextureSamplerObj	
TextureSampler wraps the OptiX C API RTtexturesampler opaque type and its associated function set	688
optix::TransformObj	
Transform wraps the OptiX C API RTtransform opaque type and its associated function set	694
optix::buffer< T, Dim >::type< T2 >	697
optix::VariableObj	
Variable object wraps OptiX C API RTvariable type and its related function set	697
optix::VectorDim< DIM >	713
optix::VectorDim< 2 >	713
optix::VectorDim< 3 >	713
optix::VectorDim< 4 >	713
optix::VectorTypes< T, Dim >	714
optix::VectorTypes< float, 1 >	714
optix::VectorTypes< float, 2 >	714
optix::VectorTypes< float, 3 >	715
optix::VectorTypes< float, 4 >	715
optix::VectorTypes< int, 1 >	716
optix::VectorTypes< int, 2 >	716
optix::VectorTypes< int, 3 >	717
optix::VectorTypes< int, 4 >	717
optix::VectorTypes< unsigned int, 1 >	718
optix::VectorTypes< unsigned int, 2 >	718
optix::VectorTypes< unsigned int, 3 >	719
optix::VectorTypes< unsigned int, 4 >	719

5 Module Documentation

5.1 OptiX API Reference

Modules

- Context handling functions
- GeometryGroup handling functions
- GroupNode functions
- SelectorNode functions
- TransformNode functions
- Acceleration functions
- GeometryInstance functions
- Geometry functions
- GeometryTriangles functions
- Material functions
- Program functions
- Buffer functions
- TextureSampler functions
- Variable functions
- CommandList functions
- Context-free functions
- CUDA C Reference
- OptiXpp wrapper
- rtu API

5.1.1 Detailed Description

OptiX API functions.

5.2 Context handling functions

Modules

- `rtContextLaunch` functions

Functions

- `RTresult RTAPI rtContextCreate (RTcontext *context)`
- `RTresult RTAPI rtContextDestroy (RTcontext context)`
- `RTresult RTAPI rtContextValidate (RTcontext context)`
- `void RTAPI rtContextGetErrorString (RTcontext context, RTresult code, const char **stringReturn)`
- `RTresult RTAPI rtContextSetAttribute (RTcontext context, RTcontextattribute attrib, RTsize size, const 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 rtContextSetStackSize (RTcontext context, RTsize bytes)`
- `RTresult RTAPI rtContextGetStackSize (RTcontext context, RTsize *bytes)`
- `RTresult RTAPI rtContextSetMaxCallableProgramDepth (RTcontext context, unsigned int maxDepth)`
- `RTresult RTAPI rtContextGetMaxCallableProgramDepth (RTcontext context, unsigned int *maxDepth)`
- `RTresult RTAPI rtContextSetMaxTraceDepth (RTcontext context, unsigned int maxDepth)`
- `RTresult RTAPI rtContextGetMaxTraceDepth (RTcontext context, unsigned int *maxDepth)`
- `RTresult RTAPI rtContextSetUsageReportCallback (RTcontext context, RTusagereportcallback callback, int verbosity, void *cbdata)`
- `RTresult RTAPI rtContextSetEntryPointCount (RTcontext context, unsigned int count)`
- `RTresult RTAPI rtContextGetEntryPointCount (RTcontext context, unsigned int *count)`
- `RTresult RTAPI rtContextSetRayGenerationProgram (RTcontext context, unsigned int entryPointIndex, RTprogram program)`
- `RTresult RTAPI rtContextGetRayGenerationProgram (RTcontext context, unsigned int entryPointIndex, RTprogram *program)`
- `RTresult RTAPI rtContextSetExceptionProgram (RTcontext context, unsigned int entryPointIndex, RTprogram program)`
- `RTresult RTAPI rtContextGetExceptionProgram (RTcontext context, unsigned int entryPointIndex, 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 rayTypeCount)`
- `RTresult RTAPI rtContextGetRayTypeCount (RTcontext context, unsigned int *rayTypeCount)`
- `RTresult RTAPI rtContextSetMissProgram (RTcontext context, unsigned int rayTypeIndex, RTprogram program)`

- RTresult RTAPI rtContextGetMissProgram (RTcontext context, unsigned int rayTypeIndex, RTprogram *program)
- RTresult RTAPI rtContextGetTextureSamplerFromId (RTcontext context, int samplerId, RTtexturesampler *sampler)
- RTresult RTAPI rtContextGetRunningState (RTcontext context, int *running)
- RTresult RTAPI rtContextLaunchProgressive2D (RTcontext context, unsigned int entryIndex, RTsize width, RTsize height, unsigned int maxSubframes)
- 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 bufferSizeBytes)
- RTresult RTAPI rtContextGetPrintBufferSize (RTcontext context, RTsize *bufferSizeBytes)
- 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 rtContextGetBufferFromId (RTcontext context, int bufferId, RTbuffer *buffer)

5.2.1 Detailed Description

Functions related to an OptiX context.

5.2.2 Function Documentation

5.2.2.1 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

5.2.2.2 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`

5.2.2.3 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](#)

5.2.2.4 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_DISK_CACHE_ENABLED` `sizeof(int)`
- `RT_CONTEXT_ATTRIBUTE_DISK_CACHE_LOCATION` `sizeof(char**)`
- `RT_CONTEXT_ATTRIBUTE_DISK_CACHE_MEMORY_LIMITS` `sizeof(RTSize[2])`

`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.

`RT_CONTEXT_ATTRIBUTE_DISK_CACHE_ENABLED` queries whether or not the OptiX disk cache is enabled.

`RT_CONTEXT_ATTRIBUTE_DISK_CACHE_LOCATION` queries the file path of the OptiX disk cache.

`RT_CONTEXT_ATTRIBUTE_DISK_CACHE_MEMORY_LIMITS` queries the low and high watermark values for the OptiX disk cache.

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`

5.2.2.5 RTresult RTAPI `rtContextGetBufferFromId` (

```
RTcontext context,
int bufferId,
RTbuffer * buffer )
```

Gets an RTbuffer corresponding to the buffer id.

Description

`rtContextGetBufferFromId` returns a handle to the buffer in **buffer* corresponding to the *bufferId* supplied. If *bufferId* 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>bufferId</i>	The ID of the buffer to query
out	<i>buffer</i>	The return handle for the buffer object corresponding to the <i>bufferId</i>

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtContextGetBufferFromId` was introduced in OptiX 3.5.

See also [rtBufferGetId](#)

5.2.2.6 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](#)

5.2.2.7 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 <code>rtContextGetDeviceCount</code>

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`

5.2.2.8 RTResult RTAPI `rtContextGetEntryPointCount` (

`RTcontext context,`
`unsigned int * count`)

Query the number of entry points for this context.

Description

`rtContextGetEntryPointCount` passes back the number of entry points associated with this context in *count*. Returns `RT_ERROR_INVALID_VALUE` if passed a *NULL* pointer.

Parameters

in	<i>context</i>	The context node to be queried
out	<i>count</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`

5.2.2.9 void RTAPI `rtContextGetErrorString` (

`RTcontext context,`
`RTResult code,`

```
const char ** stringReturn )
```

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. *stringReturn* will be set to point to this string. The memory *stringReturn* 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>stringReturn</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

5.2.2.10 RTResult RTAPI `rtContextGetExceptionEnabled` (

```
RTcontext context,
RTexception 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`, `RTexception`

5.2.2.11 RTResult RTAPI `rtContextGetExceptionProgram` (

```
RTcontext context,
unsigned int entryPointIndex,
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>entryPointIndex</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`

5.2.2.12 RTResult RTAPI `rtContextGetMaxCallableProgramDepth` (

```
RTcontext context,
unsigned int * maxDepth )
```

Query the maximum call depth for callable programs.

Description

`rtContextGetMaxCallableProgramDepth` passes back the maximum callable program call depth associated with this context in *maxDepth*. Returns `RT_ERROR_INVALID_VALUE` if passed a `NULL` pointer.

Parameters

in	<i>context</i>	The context node to be queried
out	<i>maxDepth</i>	Return parameter to store the maximum callable program depth

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtContextGetMaxCallableProgramDepth` was introduced in OptiX 6.0

See also [rtContextSetMaxCallableProgramDepth](#)

5.2.2.13 RTResult RTAPI `rtContextGetMaxTraceDepth` (

`RTcontext context,`
`unsigned int * maxDepth)`

Query the maximum trace depth for this context.

Description

`rtContextGetMaxTraceDepth` passes back the maximum trace depth associated with this context in *maxDepth*. Returns `RT_ERROR_INVALID_VALUE` if passed a *NULL* pointer.

Parameters

in	<i>context</i>	The context node to be queried
out	<i>maxDepth</i>	Return parameter to store the maximum trace depth

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtContextGetMaxTraceDepth` was introduced in OptiX 6.0.

See also [rtContextSetMaxTraceDepth](#)

5.2.2.14 RTResult RTAPI `rtContextGetMissProgram` (

`RTcontext context,`
`unsigned int rayTypeIndex,`
`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>rayTypeIndex</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`

5.2.2.15 RTResult RTAPI `rtContextGetPrintBufferSize` (

```
RTcontext context,
RTsize * bufferSizeBytes )
```

Get the current size of the print buffer.

Description

`rtContextGetPrintBufferSize` is used to query the buffer size available to hold data generated by `rtPrintf` functions. 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>bufferSizeBytes</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` functions, `rtContextSetPrintEnabled`, `rtContextGetPrintEnabled`, `rtContextSetPrintBufferSize`, `rtContextSetPrintLaunchIndex`, `rtContextGetPrintLaunchIndex`

5.2.2.16 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` functions 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` functions, `rtContextSetPrintEnabled`, `rtContextSetPrintBufferSize`, `rtContextGetPrintBufferSize`, `rtContextSetPrintLaunchIndex`, `rtContextGetPrintLaunchIndex`

5.2.2.17 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` functions 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 <i>x</i> dimension to which the output of <code>rtPrintf</code> functions invocations is limited. Will not be written to if a <code>NULL</code> pointer is passed
out	<i>y</i>	Returns the launch index in the <i>y</i> dimension to which the output of <code>rtPrintf</code> functions invocations is limited. Will not be written to if a <code>NULL</code> pointer is passed

Parameters

out	<i>z</i>	Returns the launch index in the z dimension to which the output of rtPrintf functions 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](#) functions, [rtContextGetPrintEnabled](#), [rtContextSetPrintEnabled](#), [rtContextSetPrintBufferSize](#), [rtContextGetPrintBufferSize](#), [rtContextSetPrintLaunchIndex](#)

5.2.2.18 RTResult RTAPI `rtContextGetRayGenerationProgram` (

```
RTcontext context,
unsigned int entryPointIndex,
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>entryPointIndex</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](#)

5.2.2.19 RTResult RTAPI `rtContextGetRayTypeCount` (

```
RTcontext context,
unsigned int * rayTypeCount )
```

Query the number of ray types associated with this context.

Description

`rtContextGetRayTypeCount` passes back the number of entry points associated with this context in `rayTypeCount`. Returns `RT_ERROR_INVALID_VALUE` if passed a `NULL` pointer.

Parameters

in	<code>context</code>	The context node to be queried
out	<code>rayTypeCount</code>	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`

5.2.2.20 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	<code>context</code>	The context node to be queried
out	<code>running</code>	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`

5.2.2.21 RTResult RTAPI rtContextGetStackSize (

```
RTcontext context,
RTsize * bytes )
```

Query the stack size for this context.

Description

`rtContextGetStackSize` passes back the stack size associated with this context in `bytes`. Returns `RT_ERROR_INVALID_VALUE` if passed a `NULL` pointer.

Parameters

in	<code>context</code>	The context node to be queried
out	<code>bytes</code>	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`

5.2.2.22 RTResult RTAPI rtContextGetTextureSamplerFromId (

```
RTcontext context,
int samplerId,
RTtexturesampler * sampler )
```

Gets an `RTtexturesampler` corresponding to the texture id.

Description

`rtContextGetTextureSamplerFromId` returns a handle to the texture sampler in `*sampler` corresponding to the `samplerId` supplied. If `samplerId` does not map to a valid texture handle, `*sampler` is `NULL` or if `context` is invalid, returns `RT_ERROR_INVALID_VALUE`.

Parameters

in	<code>context</code>	The context the sampler should be originated from
in	<code>samplerId</code>	The ID of the sampler to query
out	<code>sampler</code>	The return handle for the sampler object corresponding to the samplerId

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

History

`rtContextGetTextureSamplerFromId` was introduced in OptiX 3.5.

See also `rtTextureSamplerGetId`

5.2.2.23 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`

5.2.2.24 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
----	----------------	--

Parameters

<code>out</code>	<code>count</code>	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`

5.2.2.25 RTResult RTAPI `rtContextLaunchProgressive2D` (

```
RTcontext context,
unsigned int entryIndex,
RTsize width,
RTsize height,
unsigned int maxSubframes )
```

Executes a Progressive Launch for a given context.

Description

Starts the (potentially parallel) generation of subframes for progressive rendering. If *maxSubframes* 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, `Variable setters` 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>entryIndex</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>maxSubframes</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](#)

5.2.2.26 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`

5.2.2.27 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`,

5.2.2.28 RTResult RTAPI `rtContextSetAttribute` (

RTcontext *context*,
RTcontextattribute *attrib*,
RTsize *size*,
const 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_PREFER_FAST_RECOMPILES` `sizeof(int)`
- `RT_CONTEXT_ATTRIBUTE_FORCE_INLINE_USER_FUNCTIONS` `sizeof(int)`
- `RT_CONTEXT_ATTRIBUTE_DISK_CACHE_LOCATION` `sizeof(char*)`
- `RT_CONTEXT_ATTRIBUTE_DISK_CACHE_MEMORY_LIMITS` `sizeof(RTSize[2])`

`RT_CONTEXT_ATTRIBUTE_CPU_NUM_THREADS` sets the number of host CPU threads OptiX can use for various tasks.

`RT_CONTEXT_ATTRIBUTE_PREFER_FAST_RECOMPILES` is a hint about scene usage. By default OptiX produces device kernels that are optimized for the current scene. Such kernels generally run faster, but must be recompiled after some types of scene changes, causing delays. Setting `PREFER_FAST_RECOMPILES` to 1 will leave out some scene-specific optimizations, producing kernels that generally run slower but are less sensitive to changes in the scene.

`RT_CONTEXT_ATTRIBUTE_FORCE_INLINE_USER_FUNCTIONS` sets whether or not OptiX will automatically inline user functions, which is the default behavior. Please see the Programming Guide for more information about the benefits and limitations of disabling automatic inlining.

`RT_CONTEXT_ATTRIBUTE_DISK_CACHE_LOCATION` sets the location where the OptiX disk cache will be created. The location must be provided as a *NULL-terminated* string. OptiX will attempt to create the directory if it does not exist. An exception will be thrown if OptiX is unable to create the cache database file at the specified location for any reason (e.g., the path is invalid or the directory is not writable). The location of the disk cache can be overridden with the environment variable `OPTIX_CACHE_PATH`. This environment variable takes precedence over the RTcontext attribute.

`RT_CONTEXT_ATTRIBUTE_DISK_CACHE_MEMORY_LIMITS` sets the low and high watermarks for disk cache garbage collection. The limits must be passed in as a two-element array of `RTsize` values, with the low limit as the first element. OptiX will throw an exception if either limit is non-zero and the high limit is not greater than the low limit. Setting either limit to zero will disable garbage collection. Garbage collection is triggered whenever the cache data size exceeds the high watermark and proceeds until the size reaches the low watermark.

Parameters

in	<code>context</code>	The context object to be modified
in	<code>attrib</code>	Attribute to set
in	<code>size</code>	Size of the attribute being set
in	<code>p</code>	Pointer to where the value of the attribute will be copied from. This must point to at least <code>size</code> 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](#)

5.2.2.29 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](#)

5.2.2.30 RTResult RTAPI rtContextSetEntryPointCount (

```
RTcontext context,
unsigned int count )
```

Set the number of entry points for a given context.

Description

`rtContextSetEntryPointCount` sets the number of entry points associated with the given context to *count*.

Parameters

in	<i>context</i>	The context to be modified
in	<i>count</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`

5.2.2.31 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_PAYLOAD_ACCESS_OUT_OF_BOUNDS
- RT_EXCEPTION_USER_EXCEPTION_CODE_OUT_OF_BOUNDS
- RT_EXCEPTION_TRACE_DEPTH_EXCEEDED
- 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_PAYLOAD_ACCESS_OUT_OF_BOUNDS` verifies that accesses to the ray payload are within valid bounds. This exception is only supported with the RTX execution strategy.

`RT_EXCEPTION_USER_EXCEPTION_CODE_OUT_OF_BOUNDS` verifies that the exception code passed to `rtThrow` is within the valid range from `RT_EXCEPTION_USER` to `RT_EXCEPTION_USER_MAX`.

`RT_EXCEPTION_TRACE_DEPTH_EXCEEDED` verifies that the depth of the `rtTrace` tree does not exceed the limit of 31. This exception is only supported with the RTX execution strategy.

`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. This exception is supported with the RTX execution strategy only.

`RT_EXCEPTION_INVALID_RAY` checks the each ray's origin and direction values against *Nan*s 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. 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	<code>context</code>	The context for which the exception is to be enabled or disabled
in	<code>exception</code>	The exception which is to be enabled or disabled
in	<code>enabled</code>	Nonzero to enable the exception, 0 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`, `Rtexception`

5.2.2.32 RTResult RTAPI `rtContextSetExceptionProgram` (

```
RTcontext context,
unsigned int entryPointIndex,
RTprogram program )
```

Specifies the exception program for a given context entry point.

Description

`rtContextSetExceptionProgram` sets *context*'s exception program at entry point *entryPointIndex*. `RT_ERROR_INVALID_VALUE` is returned if *entryPointIndex* 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>entryPointIndex</i>	The entry point the program will be associated with
in	<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`

5.2.2.33 RTResult RTAPI `rtContextSetMaxCallableProgramDepth` (

`RTcontext context,`
`unsigned int maxDepth)`

Set maximum callable program call depth for a given context.

Description

`rtContextSetMaxCallableProgramDepth` sets the maximum call depth of a chain of callable programs for the given context to *maxDepth*. This value is only used for stack size computation. Only supported for RTX execution mode. Default value is 5. Returns `RT_ERROR_INVALID_VALUE` if context is not valid.

Parameters

in	<i>context</i>	The context node to be modified
in	<i>maxDepth</i>	The desired maximum depth

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

History

`rtContextSetMaxCallableProgramDepth` was introduced in OptiX 6.0

See also [rtContextGetMaxCallableProgramDepth](#)

5.2.2.34 RTResult RTAPI `rtContextSetMaxTraceDepth` (

RTcontext *context*,
unsigned int *maxDepth*)

Set the maximum trace depth for a given context.

Description

`rtContextSetMaxTraceDepth` sets the maximum trace depth for the given context to *maxDepth*. Only supported for RTX execution mode. Default value is 5. Maximum trace depth is 31. Returns `RT_ERROR_INVALID_VALUE` if context is not valid.

Parameters

in	<i>context</i>	The context node to be modified
in	<i>maxDepth</i>	The desired maximum depth

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

History

`rtContextSetMaxTraceDepth` was introduced in OptiX 6.0

See also [rtContextGetMaxTraceDepth](#)

5.2.2.35 RTResult RTAPI `rtContextSetMissProgram` (

RTcontext *context*,
unsigned int *rayTypeIndex*,
RTprogram *program*)

Specifies the miss program for a given context ray type.

Description

`rtContextSetMissProgram` sets *context*'s miss program associated with ray type *rayTypeIndex*. `RT_ERROR_INVALID_VALUE` is returned if *rayTypeIndex* is outside of the range [0, `rtContextGetRayTypeCount` - 1].

Parameters

in	<i>context</i>	The context node to which the miss program will be added
in	<i>rayTypeIndex</i>	The ray type the program will be associated with
in	<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`

5.2.2.36 RTResult RTAPI `rtContextSetPrintBufferSize` (

`RTcontext context,`
`RTsize bufferSizeBytes)`

Set the size of the print buffer.

Description

`rtContextSetPrintBufferSize` is used to set the buffer size available to hold data generated by `rtPrintf` functions. Returns `RT_ERROR_INVALID_VALUE` if it is called after the first invocation of `rtContextLaunch`.

Parameters

in	<code>context</code>	The context for which to set the print buffer size
in	<code>bufferSizeBytes</code>	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` functions, `rtContextSetPrintEnabled`, `rtContextGetPrintEnabled`, `rtContextGetPrintBufferSize`, `rtContextSetPrintLaunchIndex`, `rtContextGetPrintLaunchIndex`

5.2.2.37 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` functions is currently enabled for this context.

Parameters

in	<code>context</code>	The context for which printing is to be enabled or disabled
in	<code>enabled</code>	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` functions, `rtContextGetPrintEnabled`, `rtContextSetPrintBufferSize`, `rtContextGetPrintBufferSize`, `rtContextSetPrintLaunchIndex`, `rtContextGetPrintLaunchIndex`

5.2.2.38 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` functions generates output. The initial value of (x,y,z) is (-1,-1,-1), which generates output for all indices.

Parameters

in	<code>context</code>	The context for which to set the print launch index
in	<code>x</code>	The launch index in the x dimension to which to limit the output of <code>rtPrintf</code> functions invocations. If set to -1, output is generated for all launch indices in the x dimension
in	<code>y</code>	The launch index in the y dimension to which to limit the output of <code>rtPrintf</code> functions invocations. If set to -1, output is generated for all launch indices in the y dimension
in	<code>z</code>	The launch index in the z dimension to which to limit the output of <code>rtPrintf</code> functions 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` functions, `rtContextGetPrintEnabled`, `rtContextSetPrintEnabled`, `rtContextSetPrintBufferSize`, `rtContextGetPrintBufferSize`, `rtContextGetPrintLaunchIndex`

5.2.2.39 RTResult RTAPI `rtContextSetRayGenerationProgram` (

```
RTcontext context,
unsigned int entryPointIndex,
RTprogram program )
```

Specifies the ray generation program for a given context entry point.

Description

`rtContextSetRayGenerationProgram` sets *context*'s ray generation program at entry point *entryPointIndex*. `RT_ERROR_INVALID_VALUE` is returned if *entryPointIndex* 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>entryPointIndex</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`

5.2.2.40 RTResult RTAPI `rtContextSetRayTypeCount` (

```
RTcontext context,
unsigned int rayTypeCount )
```

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>rayTypeCount</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](#)

5.2.2.41 RTResult RTAPI `rtContextSetStackSize` (

`RTcontext context,`
`RTsize bytes)`

Set the stack size for a given context.

Description

`rtContextSetStackSize` sets the stack size for the given context to *bytes* bytes. Not supported with the RTX execution strategy. With RTX execution strategy `rtContextSetMaxTraceDepth` and `rtContextSetMaxCallableProgramDepth` should be used to control stack size. Returns `RT_ERROR_INVALID_VALUE` if context is not valid.

Parameters

in	<i>context</i>	The context node to be modified
in	<i>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](#)

5.2.2.42 RTResult RTAPI `rtContextSetUsageReportCallback` (

`RTcontext context,`
`RTusagereportcallback callback,`

```
int verbosity,
void * cbdata )
```

Set usage report callback function.

Description

`rtContextSetUsageReportCallback` sets an application-side callback function *callback* and a verbosity level *verbosity*.

`RTusagereportcallback` is defined as `void (RTusagereportcallback)(int, const char, const char*, void*)`.

The provided callback will be invoked with the message's verbosity level as the first parameter. The second parameter is a descriptive tag string and the third parameter is the message itself. The fourth parameter is a pointer to user-defined data, which may be NULL. The descriptive tag will give a terse message category description (eg, 'SCENE STAT'). The messages will be unstructured and subject to change with subsequent releases. The verbosity argument specifies the granularity of these messages.

verbosity of 0 disables reporting. *callback* is ignored in this case.

verbosity of 1 enables error messages and important warnings. This verbosity level can be expected to be efficient and have no significant overhead.

verbosity of 2 additionally enables minor warnings, performance recommendations, and scene statistics at startup or recompilation granularity. This level may have a performance cost.

verbosity of 3 additionally enables informational messages and per-launch statistics and messages.

A NULL *callback* when *verbosity* is non-zero or a *verbosity* outside of [0, 3] will result in

`RT_ERROR_INVALID_VALUE` return code.

Only one report callback function can be specified at any time.

Parameters

in	<i>context</i>	The context node to be modified
in	<i>callback</i>	The function to be called
in	<i>verbosity</i>	The verbosity of report messages
in	<i>cbdata</i>	Pointer to user-defined data that will be sent to the callback. Can be NULL.

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

History

`rtContextSetUsageReportCallback` was introduced in OptiX 5.0.

See also

5.2.2.43 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

in	<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](#)

5.2.2.44 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

in	<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 `rtContextGetString`

5.3 rtContextLaunch functions

Functions

- RTresult RTAPI rtContextLaunch1D (RTcontext context, unsigned int entryPointIndex, RTsize width)
- RTresult RTAPI rtContextLaunch2D (RTcontext context, unsigned int entryPointIndex, RTsize width, RTsize height)
- RTresult RTAPI rtContextLaunch3D (RTcontext context, unsigned int entryPointIndex, RTsize width, RTsize height, RTsize depth)

5.3.1 Detailed Description

Functions designed to launch OptiX ray tracing.

5.3.2 Function Documentation

5.3.2.1 RTresult RTAPI rtContextLaunch1D (
 RTcontext *context*,
 unsigned int *entryPointIndex*,
 RTsize *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 `entryPointIndex` will be invoked once for every element (pixel or voxel) of the computation grid specified by `width`, `height`, and `depth`.

For 3D launches, the product of `width` and `depth` must be smaller than 4294967296 (2^{32}).

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>entryPointIndex</i>	The initial entry point into kernel
in	<i>width</i>	Width of the computation grid

5.3.2.2 RTResult RTAPI rtContextLaunch2D (

```
RTcontext context,  

unsigned int entryPointIndex,  

RTsize width,  

RTsize height )
```

Parameters

in	<i>context</i>	The context to be executed
in	<i>entryPointIndex</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

5.3.2.3 RTResult RTAPI rtContextLaunch3D (

```
RTcontext context,  

unsigned int entryPointIndex,  

RTsize width,  

RTsize height,  

RTsize depth )
```

Parameters

in	<i>context</i>	The context to be executed
in	<i>entryPointIndex</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>depth</i>	Depth of the computation grid

5.4 GeometryGroup handling functions

Functions

- 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 rtGeometryGroupSetFlags (RTgeometrygroup group, RTinstanceflags flags)
- RTresult RTAPI rtGeometryGroupGetFlags (RTgeometrygroup group, RTinstanceflags *flags)
- RTresult RTAPI rtGeometryGroupSetVisibilityMask (RTgeometrygroup group, RTvisibilitymask mask)
- RTresult RTAPI rtGeometryGroupGetVisibilityMask (RTgeometrygroup group, RTvisibilitymask *mask)
- 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)

5.4.1 Detailed Description

Functions related to an OptiX Geometry Group node.

5.4.2 Function Documentation

5.4.2.1 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`

5.4.2.2 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`

5.4.2.3 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	<code>geometrygroup</code>	The geometry group handle
out	<code>acceleration</code>	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`

5.4.2.4 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	<code>geometrygroup</code>	The parent geometry group handle
in	<code>index</code>	The index of the child slot to query
out	<code>geometryinstance</code>	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`,

5.4.2.5 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`

5.4.2.6 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`

5.4.2.7 RTresult RTAPI `rtGeometryGroupGetFlags` (

RTgeometrygroup *group*,
RTinstanceflags * *flags*)

Gets instance flags of a geometry group.

Description

See `rtGeometryGroupSetFlags` for details.

Parameters

in	<i>group</i>	The group handle
out	<i>flags</i>	Instance flags for the given geometry group

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtGeometryGroupGetFlags` was introduced in OptiX 6.0.

See also `rtGeometryGroupSetFlags`, `rtTrace`

5.4.2.8 RTresult RTAPI `rtGeometryGroupGetVisibilityMask` (

RTgeometrygroup *group*,

RTvisibilitymask * mask)

Gets the visibility mask of a geometry group.

Description See [rtGeometryGroupSetVisibilityMask](#) for details/

Parameters

in	<i>group</i>	The group handle
out	<i>mask</i>	A set of bits for which rays will intersect the group

Return values

Relevant return values:

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

History

[rtGeometryGroupGetVisibilityMask](#) was introduced in OptiX 6.0.

See also [rtGroupGetVisibilityMask](#) [rtGeometryGroupSetVisibilityMask](#), [rtTrace](#)

5.4.2.9 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`

5.4.2.10 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`

5.4.2.11 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`

5.4.2.12 RTResult RTAPI `rtGeometryGroupSetFlags` (

`RTgeometrygroup group,`
`RTinstanceflags flags)`

Sets instance flags for a geometry group.

Description

This function controls the `RTinstanceflags` of the given geometry group. The flags override the `RTgeometryflags` of the underlying geometry where appropriate.

Parameters

in	<i>group</i>	The group handle
in	<i>flags</i>	Instance flags for the given geometry group

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

History

`rtGeometryGroupSetFlags` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetFlagsPerMaterial`, `rtGeometrySetFlags`, `rtGeometryGroupGetFlags`, `rtTrace`

5.4.2.13 RTResult RTAPI rtGeometryGroupSetVisibilityMask (

RTgeometrygroup *group*,
RTvisibilitymask *mask*)

Sets the visibility mask of a geometry group.

Description Geometry is intersected by rays if the ray's RTvisibilitymask shares at least one bit with the geometry's mask. This mechanism allows for a number of user-defined visibility groups that can be excluded from certain types of rays as needed.

Note that the

Precondition

mask is currently limited to 8 bits.

Parameters

in	<i>group</i>	The group handle
in	<i>mask</i>	A set of bits for which rays will intersect the group

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

rtGeometryGroupSetVisibilityMask was introduced in OptiX 6.0.

See also [rtGroupSetVisibilityMask](#) [rtGeometryGroupGetVisibilityMask](#), [rtTrace](#)

5.4.2.14 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](#)

5.5 GroupNode functions

Functions

- 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 rtGroupSetVisibilityMask (RTgroup group, RTvisibilitymask mask)
- RTresult RTAPI rtGroupGetVisibilityMask (RTgroup group, RTvisibilitymask *mask)
- 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)

5.5.1 Detailed Description

Functions related to an OptiX Group node.

5.5.2 Function Documentation

5.5.2.1 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`

5.5.2.2 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`

5.5.2.3 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`

5.5.2.4 RTResult RTAPI `rtGroupGetChild` (

```
RTgroup group,
unsigned int index,
RTobject * 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`

5.5.2.5 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`

5.5.2.6 RTresult RTAPI `rtGroupGetChildType` (

```
RTgroup group,
unsigned int index,
RTobjecttype * 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`

5.5.2.7 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`

5.5.2.8 RTresult RTAPI rtGroupGetVisibilityMask (

RTgroup *group*,
RTvisibilitymask * *mask*)

Retrieves the visibility mask of a group.

Description See `rtGroupSetVisibilityMask` for details.

Parameters

in	<i>group</i>	The group handle
out	<i>mask</i>	A set of bits for which rays will intersect the group

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtGroupGetVisibilityMask` was introduced in OptiX 6.0.

See also `rtGeometryGroupGetVisibilityMask`, `rtGroupSetVisibilityMask`, `rtTrace`

5.5.2.9 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`

5.5.2.10 RTresult RTAPI `rtGroupSetChild` (

RTgroup *group*,
unsigned int *index*,
RTobject *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 { <code>RTgroup</code> , <code>RTselector</code> , <code>RTgeometrygroup</code> , <code>RTtransform</code> }

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`

5.5.2.11 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`

5.5.2.12 RTResult RTAPI rtGroupSetVisibilityMask (

RTgroup *group*,
RTvisibilitymask *mask*)

Sets the visibility mask for a group.

Description Geometry is intersected by rays if the ray's **RTvisibilitymask** shares at least one bit with the geometry's mask. This mechanism allows for a number of user-defined visibility groups that can be excluded from certain types of rays as needed.

Note that the

Precondition

mask is currently limited to 8 bits.

Parameters

in	<i>group</i>	The group handle
in	<i>mask</i>	A set of bits for which rays will intersect the group

Return values

Relevant return values:

- **RT_SUCCESS**
- **RT_ERROR_INVALID_VALUE**

History

`rtGroupSetVisibilityMask` was introduced in OptiX 6.0.

See also [rtGeometryGroupSetVisibilityMask](#), [rtGroupGetVisibilityMask](#), [rtTrace](#)

5.5.2.13 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`

5.6 SelectorNode functions

Functions

- 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)

5.6.1 Detailed Description

Functions related to an OptiX Selector node.

5.6.2 Function Documentation

5.6.2.1 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 an invalid state. For the node to be valid, 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`

5.6.2.2 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`, `Variable setters{...}`

5.6.2.3 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	<code>selector</code>	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`

5.6.2.4 RTResult RTAPI rtSelectorGetChild (

RTselector selector,
unsigned int index,
RTobject * 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 `RTobject` 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	<code>selector</code>	Selector node handle
in	<code>index</code>	Child node index
out	<code>child</code>	Child node handle. Can be { <code>RTgroup</code> , <code>RTselector</code> , <code>RTgeometrygroup</code> , <code>RTtransform</code> }

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`

5.6.2.5 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`

5.6.2.6 RTResult RTAPI `rtSelectorGetChildType` (

```
RTselector selector,
unsigned int index,
RTobjecttype * 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`

5.6.2.7 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`

5.6.2.8 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{...}`

5.6.2.9 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](#)

5.6.2.10 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](#)

5.6.2.11 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{...}`

5.6.2.12 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`

5.6.2.13 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 { <code>RTgroup</code> , <code>RTselector</code> , <code>RTgeometrygroup</code> , <code>RTtransform</code> }

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`

5.6.2.14 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`

5.6.2.15 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`

5.6.2.16 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`

5.7 TransformNode functions

Functions

- 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 *inverseMatrix)
- RTresult RTAPI rtTransformGetMatrix (RTtransform transform, int transpose, float *matrix, float *inverseMatrix)
- RTresult RTAPI rtTransformSetMotionRange (RTtransform transform, float timeBegin, float timeEnd)
- RTresult RTAPI rtTransformGetMotionRange (RTtransform transform, float *timeBegin, float *timeEnd)
- RTresult RTAPI rtTransformSetMotionBorderMode (RTtransform transform, RTmotionbordermode beginMode, RTmotionbordermode endMode)
- RTresult RTAPI rtTransformGetMotionBorderMode (RTtransform transform, RTmotionbordermode *beginMode, RTmotionbordermode *endMode)
- RTresult RTAPI rtTransformSetMotionKeys (RTtransform transform, unsigned int n, RTmotionkeytype type, const float *keys)
- RTresult RTAPI rtTransformGetMotionKeyType (RTtransform transform, RTmotionkeytype *type)
- RTresult RTAPI rtTransformGetMotionKeyCount (RTtransform transform, unsigned int *n)
- RTresult RTAPI rtTransformGetMotionKeys (RTtransform transform, float *keys)
- RTresult RTAPI rtTransformSetChild (RTtransform transform, RTobject child)
- RTresult RTAPI rtTransformGetChild (RTtransform transform, RTobject *child)
- RTresult RTAPI rtTransformGetChildType (RTtransform transform, RTobjecttype *type)

5.7.1 Detailed Description

Functions related to an OptiX Transform node.

5.7.2 Function Documentation

5.7.2.1 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`

5.7.2.2 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`

5.7.2.3 RTresult RTAPI `rtTransformGetChild` (

RTtransform *transform*,

RTobject * 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 `RTobject` 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 { <code>RTgroup</code> , <code>RTselector</code> , <code>RTgeometrygroup</code> , <code>RTtransform</code> }

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`

5.7.2.4 RTResult RTAPI `rtTransformGetChildType` (

`RTtransform transform,`
`RTobjecttype * 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`

5.7.2.5 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`

5.7.2.6 RTResult RTAPI `rtTransformGetMatrix` (

`RTtransform transform,`
`int transpose,`
`float * matrix,`

```
float * inverseMatrix )
```

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 *inverseMatrix*. 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>inverseMatrix</i> should be transposed
out	<i>matrix</i>	Affine matrix (4x4 float array)
out	<i>inverseMatrix</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`

5.7.2.7 RTresult RTAPI `rtTransformGetMotionBorderMode (`

```
RTtransform transform,
RTmotionbordermode * beginMode,
RTmotionbordermode * endMode )
```

Returns the motion border modes of a Transform node.

Description `rtTransformGetMotionBorderMode` returns the motion border modes for the time range associated with *transform*.

Parameters

in	<i>transform</i>	Transform node handle
out	<i>beginMode</i>	Motion border mode at motion time range begin
out	<i>endMode</i>	Motion border mode at motion time range end

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtTransformGetMotionBorderMode` was introduced in OptiX 5.0.

See also `rtTransformSetMotionBorderMode`, `rtTransformGetMotionRange`, `rtTransformGetMotionKeyCount`, `rtTransformGetMotionKeyType`, `rtTransformGetMotionKeys`,

5.7.2.8 RTresult RTAPI `rtTransformGetMotionKeyCount` (

`RTtransform transform,`
`unsigned int * n)`

Returns the number of motion keys associated with a Transform node.

Description `rtTransformGetMotionKeyCount` returns in `n` the number of motion keys associated with `transform` using `rtTransformSetMotionKeys`. Note that the default value is 1, not 0, for a transform without motion.

Parameters

in	<code>transform</code>	Transform node handle
out	<code>n</code>	Number of motion steps <code>n</code> ≥ 1

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtTransformGetMotionKeyCount` was introduced in OptiX 5.0.

See also `rtTransformSetMotionKeys`, `rtTransformGetMotionBorderMode`, `rtTransformGetMotionRange`, `rtTransformGetMotionKeyType` `rtTransformGetMotionKeys`

5.7.2.9 RTresult RTAPI `rtTransformGetMotionKeys` (

`RTtransform transform,`
`float * keys)`

Returns the motion keys associated with a Transform node.

Description `rtTransformGetMotionKeys` returns in `keys` packed float values for all motion keys. The `keys` array must be large enough to hold all the keys, based on the key type returned by

`rtTransformGetMotionKeyType` and the number of keys returned by `rtTransformGetMotionKeyCount`. A single key consists of either 12 floats (type `RT_MOTIONKEYTYPE_MATRIX_FLOAT12`) or 16 floats (type `RT_MOTIONKEYTYPE_SRT_FLOAT16`).

Parameters

in	<i>transform</i>	Transform node handle
out	<i>keys</i>	Motion keys associated with this Transform

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

History

`rtTransformGetMotionKeys` was introduced in OptiX 5.0.

See also `rtTransformSetMotionKeys`, `rtTransformGetMotionBorderMode`, `rtTransformGetMotionRange`, `rtTransformGetMotionKeyCount`, `rtTransformGetMotionKeyType`

5.7.2.10 RTResult RTAPI `rtTransformGetMotionKeyType` (
RTtransform *transform*,
RTmotionkeytype * *type*)

Returns the motion key type associated with a Transform node.

Description `rtTransformGetMotionKeyType` returns the key type from the most recent call to `rtTransformSetMotionKeys`, or `RT_MOTIONKEYTYPE_NONE` if no keys have been set.

Parameters

in	<i>transform</i>	Transform node handle
out	<i>type</i>	Motion key type associated with this Transform

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

History

`rtTransformGetMotionKeyType` was introduced in OptiX 5.0.

See also `rtTransformSetMotionKeys`, `rtTransformGetMotionBorderMode`, `rtTransformGetMotionRange`, `rtTransformGetMotionKeyCount`, `rtTransformGetMotionKeys`

5.7.2.11 RTResult RTAPI rtTransformGetMotionRange (

RTtransform *transform*,

float * *timeBegin*,

float * *timeEnd*)

Returns the motion time range associated with a Transform node.

Description `rtTransformGetMotionRange` returns the motion time range set for the Transform.

Parameters

in	<i>transform</i>	Transform node handle
out	<i>timeBegin</i>	Beginning time value of range
out	<i>timeEnd</i>	Ending time value of range

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

History

`rtTransformGetMotionRange` was introduced in OptiX 5.0.

See also `rtTransformSetMotionRange`, `rtTransformGetMotionBorderMode`, `rtTransformGetMotionKeyCount`, `rtTransformGetMotionKeyType`, `rtTransformGetMotionKeys`,

5.7.2.12 RTResult RTAPI rtTransformSetChild (

RTtransform *transform*,

RTobject *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 { <code>RTgroup</code> , <code>RTselector</code> , <code>RTgeometrygroup</code> , <code>RTtransform</code> }

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`

5.7.2.13 RTResult RTAPI `rtTransformSetMatrix` (

```

RTtransform transform,
int transpose,
const float * matrix,
const float * inverseMatrix )
```

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 *inverseMatrix* 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 *a14*, *a24*, and *a34* 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.

Calling this function clears any motion keys previously set for the Transform.

Parameters

in	<i>transform</i>	Transform node handle
in	<i>transpose</i>	Flag indicating whether <i>matrix</i> and <i>inverseMatrix</i> should be transposed
in	<i>matrix</i>	Affine matrix (4x4 float array)
in	<i>inverseMatrix</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](#)

5.7.2.14 RTResult RTAPI `rtTransformSetMotionBorderMode` (

`RTtransform transform,`
`RTmotionbordermode beginMode,`
`RTmotionbordermode endMode)`

Sets the motion border modes of a Transform node.

Description `rtTransformSetMotionBorderMode` sets the behavior of *transform* outside its motion time range. The *beginMode* and *endMode* arguments correspond to *timeBegin* and *timeEnd* set with `rtTransformSetMotionRange`. The arguments are independent, and each has one of the following values:

- `RT_MOTIONBORDERMODE_CLAMP` : The transform and the scene under it still exist at times less than *timeBegin* or greater than *timeEnd*, with the transform clamped to its values at *timeBegin* or *timeEnd*, respectively.
- `RT_MOTIONBORDERMODE_VANISH` : The transform and the scene under it vanish for times less than *timeBegin* or greater than *timeEnd*.

Parameters

in	<i>transform</i>	Transform node handle
in	<i>beginMode</i>	Motion border mode at motion range begin
in	<i>endMode</i>	Motion border mode at motion range end

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

History

`rtTransformSetMotionBorderMode` was introduced in OptiX 5.0.

See also [rtTransformGetMotionBorderMode](#), [rtTransformSetMotionRange](#), [rtTransformSetMotionKeys](#),

5.7.2.15 RTResult RTAPI `rtTransformSetMotionKeys` (

`RTtransform transform,`
`unsigned int n,`
`RTmotionkeytype type,`
`const float * keys)`

Sets the motion keys associated with a Transform node.

Description `rtTransformSetMotionKeys` sets a series of key values defining how *transform* varies with time. The float values in *keys* are one of the following types:

- `RT_MOTIONKEYTYPE_MATRIX_FLOAT12` Each key is a 12-float 3x4 matrix in row major order (3 rows, 4 columns). The length of *keys* is 12*n.
- `RT_MOTIONKEYTYPE_SRT_FLOAT16` Each key is a packed 16-float array in this order: [sx, a, b, pvx, sy, c, pvy, sz, pvz, qx, qy, qz, qw, tx, ty, tz] The length of *keys* is 16*n.

These are packed components of a scale/shear S, a quaternion R, and a translation T.

$$S = [sx \ a \ b \ pvx] [* \ sy \ c \ pvy] [* \ * \ sz \ pvz]$$

$$R = [qx, qy, qz, qw] \text{ where } qw = \cos(\theta/2) \text{ and } [qx, qy, qz] = \sin(\theta/2) * \text{normalized_axis}.$$

$$T = [tx, ty, tz]$$

Removing motion keys:

Passing a single key with *n* == 1, or calling `rtTransformSetMatrix`, removes any motion data from *transform*, and sets its matrix to values derived from the single key.

Parameters

in	<i>transform</i>	Transform node handle
in	<i>n</i>	Number of motion keys >= 1
in	<i>type</i>	Type of motion keys
in	<i>keys</i>	<i>n</i> Motion keys associated with this Transform

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

History

`rtTransformSetMotionKeys` was introduced in OptiX 5.0.

See also `rtTransformGetMotionKeyCount`, `rtTransformGetMotionKeyType`, `rtTransformGetMotionKeys`, `rtTransformSetMotionBorderMode`, `rtTransformSetMotionRange`,

5.7.2.16 RTresult RTAPI `rtTransformSetMotionRange` (

```
RTtransform transform,
float timeBegin,
float timeEnd )
```

Sets the motion time range for a Transform node.

Description Sets the inclusive motion time range [*timeBegin*, *timeEnd*] for *transform*, where *timeBegin* <= *timeEnd*. The default time range is [0.0, 1.0]. Has no effect unless `rtTransformSetMotionKeys` is also called, in which case the left endpoint of the time range, *timeBegin*, is associated with the first

motion key, and the right endpoint, *timeEnd*, with the last motion key. The keys uniformly divide the time range.

Parameters

in	<i>transform</i>	Transform node handle
in	<i>timeBegin</i>	Beginning time value of range
in	<i>timeEnd</i>	Ending time value of range

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtTransformSetMotionRange` was introduced in OptiX 5.0.

See also `rtTransformGetMotionRange`, `rtTransformSetMotionBorderMode`, `rtTransformSetMotionKeys`,

5.7.2.17 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](#)

5.8 Acceleration functions

Functions

- 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 **stringReturn)
- RTresult RTAPI rtAccelerationSetProperty (RTacceleration acceleration, const char *name, const char *value)
- RTresult RTAPI rtAccelerationGetProperty (RTacceleration acceleration, const char *name, const char **stringReturn)
- RTresult RTAPI rtAccelerationMarkDirty (RTacceleration acceleration)
- RTresult RTAPI rtAccelerationIsDirty (RTacceleration acceleration, int *dirty)

5.8.1 Detailed Description

Functions related to an OptiX Acceleration Structure node.

5.8.2 Function Documentation

5.8.2.1 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	<code>context</code>	Specifies a context within which to create a new acceleration structure
out	<code>acceleration</code>	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`

5.8.2.2 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`

5.8.2.3 RTResult RTAPI `rtAccelerationGetBuilder` (*RTacceleration acceleration*, *const char ** stringReturn*)

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. *stringReturn* will be set to point to the returned string. The memory *stringReturn* 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>stringReturn</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`

5.8.2.4 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`

5.8.2.5 RTResult RTAPI `rtAccelerationGetProperty` (

`RTacceleration acceleration,`
`const char * name,`
`const char ** stringReturn)`

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. *stringReturn* will be set to point to the returned string. The memory *stringReturn* 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>stringReturn</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`,

5.8.2.6 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` functions

5.8.2.7 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 `rtAccelerationIsDirty`, `rtContextLaunch` functions

5.8.2.8 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. Supports motion blur.
- "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`

5.8.2.9 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: Trbvh, 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: Trbvh, Sbvh 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: Trbvh, Sbvh 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: Trbvh, Sbvh 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: Trbvh, Sbvh 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: Trbvh 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 Trbvh but can result in slower rendering performance.
- "motion_steps" Available in: Trbvh Number of motion steps to build into an acceleration structure that contains motion geometry or motion transforms. Ignored for acceleration structures built over static nodes. Gives a tradeoff between device memory and time: if the input geometry or transforms have many motion steps, then increasing the motion steps in the acceleration structure may result in faster traversal, at the cost of linear increase in memory usage. Default 2, and clamped >=1.

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](#),

5.8.2.10 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`

5.9 GeometryInstance functions

Functions

- 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 rtGeometryInstanceSetGeometryTriangles (RTgeometryinstance geometryinstance, RTgeometrytriangles geometrytriangles)
- RTresult RTAPI rtGeometryInstanceGetGeometryTriangles (RTgeometryinstance geometryinstance, RTgeometrytriangles *geometrytriangles)
- 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)

5.9.1 Detailed Description

Functions related to an OptiX Geometry Instance node.

5.9.2 Function Documentation

5.9.2.1 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`

5.9.2.2 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 Variable setters functions. Once a variable is set, its type cannot be changed anymore.

Parameters

in	<i>geometryinstance</i>	Specifies the associated GeometryInstance node
----	-------------------------	--

Parameters

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 [Variable functions](#), [rtGeometryInstanceQueryVariable](#), [rtGeometryInstanceGetVariable](#), [rtGeometryInstanceRemoveVariable](#)

5.9.2.3 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](#)

5.9.2.4 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`

5.9.2.5 RTresult RTAPI `rtGeometryInstanceGetGeometry` (

```
RTgeometryinstance geometryinstance,
RTgeometry * geometry )
```

Returns the attached Geometry node.

Description

`rtGeometryInstanceGetGeometry` sets *geometry* to the handle of the attached Geometry node. Only one GeometryTriangles or Geometry node can be attached to a GeometryInstance at a time.

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` `rtGeometryInstanceSetGeometryTriangles`
`rtGeometryInstanceGetGeometryTriangles`

5.9.2.6 RTresult RTAPI `rtGeometryInstanceGetGeometryTriangles` (

`RTgeometryinstance geometryinstance,`
`RTgeometrytriangles * geometrytriangles`)

Returns the attached Geometry node.

Description

`rtGeometryInstanceGetGeometryTriangles` sets `geometrytriangles` to the handle of the attached GeometryTriangles node. If no GeometryTriangles node is attached or a Geometry node is attached, `RT_ERROR_INVALID_VALUE` is returned, else `RT_SUCCESS`.

Parameters

in	<code>geometryinstance</code>	GeometryInstance node handle to query geometrytriangles
out	<code>geometrytriangles</code>	Handle to attached GeometryTriangles node

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`
- `RT_ERROR_MEMORY_ALLOCATION_FAILED`

History

`rtGeometryInstanceGetGeometryTriangles` was introduced in OptiX 6.0.

See also `rtGeometryInstanceCreate`, `rtGeometryInstanceDestroy`, `rtGeometryInstanceValidate`,
`rtGeometryInstanceSetGeometryTriangles` `rtGeometryInstanceSetGeometry`
`rtGeometryInstanceGetGeometry`

5.9.2.7 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`

5.9.2.8 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 materials 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`

5.9.2.9 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`

5.9.2.10 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`

5.9.2.11 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`

5.9.2.12 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	<code>geometryinstance</code>	The GeometryInstance node from which to remove a variable
in	<code>v</code>	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`

5.9.2.13 RTResult RTAPI `rtGeometryInstanceSetGeometry` (

`RTgeometryinstance geometryinstance,`
`RTgeometry geometry)`

Attaches a Geometry node.

Description

`rtGeometryInstanceSetGeometry` attaches a Geometry node to a GeometryInstance. Only one GeometryTriangles or Geometry node can be attached to a GeometryInstance at a time. However, it is possible at any time to attach a different GeometryTriangles or Geometry via `rtGeometryInstanceSetGeometryTriangles` or `rtGeometryInstanceSetGeometry` respectively.

Parameters

in	<i>geometryinstance</i>	GeometryInstance node handle to attach <i>geometry</i> to
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` `rtGeometryInstanceGetGeometryTriangles`
`rtGeometryInstanceSetGeometryTriangles`

5.9.2.14 RTResult RTAPI rtGeometryInstanceSetGeometryTriangles (
RTgeometryinstance *geometryinstance*,
RTgeometrytriangles *geometrytriangles*)

Attaches a Geometry node.

Description

`rtGeometryInstanceSetGeometryTriangles` attaches a `GeometryTriangles` node to a `GeometryInstance`. Only one `GeometryTriangles` or `Geometry` node can be attached to a `GeometryInstance` at a time. However, it is possible at any time to attach a different `GeometryTriangles` or `Geometry` via `rtGeometryInstanceSetGeometryTriangles` or `rtGeometryInstanceSetGeometry` respectively.

Parameters

in	<i>geometryinstance</i>	GeometryInstance node handle to attach <i>geometrytriangles</i> to
in	<i>geometrytriangles</i>	GeometryTriangles 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

`rtGeometryInstanceSetGeometryTriangles` was introduced in OptiX 6.0.

See also [rtGeometryInstanceGetGeometryTriangles](#) [rtGeometryInstanceSetGeometry](#)
[rtGeometryInstanceGetGeometry](#)

5.9.2.15 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](#)

5.9.2.16 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](#)

5.9.2.17 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	<code>geometryinstance</code>	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](#)

5.10 Geometry functions

Functions

- 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 primitiveCount)
- RTresult RTAPI rtGeometryGetPrimitiveCount (RTgeometry geometry, unsigned int *primitiveCount)
- RTresult RTAPI rtGeometrySetPrimitiveIndexOffset (RTgeometry geometry, unsigned int indexOffset)
- RTresult RTAPI rtGeometryGetPrimitiveIndexOffset (RTgeometry geometry, unsigned int *indexOffset)
- RTresult RTAPI rtGeometrySetMotionRange (RTgeometry geometry, float timeBegin, float timeEnd)
- RTresult RTAPI rtGeometryGetMotionRange (RTgeometry geometry, float *timeBegin, float *timeEnd)
- RTresult RTAPI rtGeometrySetMotionBorderMode (RTgeometry geometry, RTmotionbordermode beginMode, RTmotionbordermode endMode)
- RTresult RTAPI rtGeometryGetMotionBorderMode (RTgeometry geometry, RTmotionbordermode *beginMode, RTmotionbordermode *endMode)
- RTresult RTAPI rtGeometrySetMotionSteps (RTgeometry geometry, unsigned int n)
- RTresult RTAPI rtGeometryGetMotionSteps (RTgeometry geometry, unsigned int *n)
- 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 rtGeometrySetFlags (RTgeometry geometry, RTgeometryflags flags)
- RTresult RTAPI rtGeometryGetFlags (RTgeometry geometry, RTgeometryflags *flags)
- 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)

5.10.1 Detailed Description

Functions related to an OptiX Geometry node.

5.10.2 Function Documentation

5.10.2.1 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	<i>context</i>	Specifies the rendering context of the Geometry node
out	<i>geometry</i>	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`

5.10.2.2 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 `Variable` setters 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 [Variable functions](#), [rtGeometryQueryVariable](#), [rtGeometryGetVariable](#), [rtGeometryRemoveVariable](#)

5.10.2.3 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](#)

5.10.2.4 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`

5.10.2.5 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`

5.10.2.6 RTResult RTAPI `rtGeometryGetFlags` (
RTgeometry *geometry*,
RTgeometryflags * *flags*)

Retrieves geometry flags.

Description

See `rtGeometrySetFlags` for details.

Parameters

in	<i>geometry</i>	The group handle
out	<i>flags</i>	Flags for the given geometry group

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtGeometryGetFlags` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetFlagsPerMaterial`, `rtTrace`

5.10.2.7 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`

5.10.2.8 RTResult RTAPI `rtGeometryGetMotionBorderMode` (

```
RTgeometry geometry,
RTmotionbordermode * beginMode,
RTmotionbordermode * endMode )
```

Returns the motion border modes of a Geometry node.

Description `rtGeometryGetMotionBorderMode` returns the motion border modes for the time range associated with *geometry*.

Parameters

in	<i>geometry</i>	Geometry node handle
out	<i>beginMode</i>	Motion border mode at motion range begin
out	<i>endMode</i>	Motion border mode at motion range end

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryGetMotionBorderMode` was introduced in OptiX 5.0.

See also `rtGeometrySetMotionBorderMode` `rtGeometryGetMotionRange` `rtGeometryGetMotionSteps`

5.10.2.9 RTResult RTAPI `rtGeometryGetMotionRange` (

```
RTgeometry geometry,
float * timeBegin,
float * timeEnd )
```

Returns the motion time range associated with a Geometry node.

Description `rtGeometryGetMotionRange` returns the motion time range associated with *geometry* from a previous call to `rtGeometrySetMotionRange`, or the default values of [0.0, 1.0].

Parameters

in	<i>geometry</i>	Geometry node handle
out	<i>timeBegin</i>	Beginning time value of range
out	<i>timeEnd</i>	Ending time value of range

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

History

`rtGeometryGetMotionRange` was introduced in OptiX 5.0.

See also `rtGeometrySetMotionRange` `rtGeometryGetMotionBorderMode` `rtGeometryGetMotionSteps`

5.10.2.10 RTResult RTAPI `rtGeometryGetMotionSteps` (

`RTgeometry geometry,`
`unsigned int * n)`

Returns the number of motion steps associated with a Geometry node.

Description `rtGeometryGetMotionSteps` returns in *n* the number of motion steps associated with *geometry*. Note that the default value is 1, not 0, for geometry without motion.

Parameters

in	<i>geometry</i>	Geometry node handle
out	<i>n</i>	Number of motion steps <i>n</i> ≥ 1

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

History

`rtGeometryGetMotionSteps` was introduced in OptiX 5.0.

See also `rtGeometryGetMotionSteps` `rtGeometrySetMotionBorderMode` `rtGeometrySetMotionRange`

5.10.2.11 RTResult RTAPI `rtGeometryGetPrimitiveCount` (

```
RTgeometry geometry,
unsigned int * primitiveCount )
```

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>primitiveCount</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`

5.10.2.12 RTResult RTAPI `rtGeometryGetPrimitiveIndexOffset` (

```
RTgeometry geometry,
unsigned int * indexOffset )
```

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>indexOffset</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`

5.10.2.13 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`, `rtGeometryQueryVariable`

5.10.2.14 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`

5.10.2.15 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`

5.10.2.16 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`

5.10.2.17 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.

If `geometry` has more than one motion step, set using `rtGeometrySetMotionSteps`, then the bounding box program must compute a bounding box per primitive and per motion step.

Parameters

in	<code>geometry</code>	The geometry node for which to set the bounding box program
in	<code>program</code>	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`

5.10.2.18 RTResult RTAPI rtGeometrySetFlags (
RTgeometry *geometry*,
RTgeometryflags *flags*)

Sets geometry flags.

Description

See `rtGeometryTrianglesSetFlagsPerMaterial` for a description of the behavior of the various flags.

Parameters

in	<code>geometry</code>	The group handle
out	<code>flags</code>	Flags for the given geometry group

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

History

`rtGeometrySetFlags` was introduced in OptiX 6.0.

See also [rtGeometryTrianglesSetFlagsPerMaterial](#), [rtTrace](#)

5.10.2.19 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](#)

5.10.2.20 RTResult RTAPI rtGeometrySetMotionBorderMode (

RTgeometry *geometry*,

RTmotionbordermode *beginMode*,

RTmotionbordermode *endMode*)

Sets the motion border modes of a Geometry node.

Description `rtGeometrySetMotionBorderMode` sets the behavior of *geometry* outside its motion time range. Options are `RT_MOTIONBORDERMODE_CLAMP` or `RT_MOTIONBORDERMODE_VANISH`. See [rtTransformSetMotionBorderMode](#) for details.

Parameters

in	<i>geometry</i>	Geometry node handle
in	<i>beginMode</i>	Motion border mode at motion range begin
in	<i>endMode</i>	Motion border mode at motion range end

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometrySetMotionBorderMode` was introduced in OptiX 5.0.

See also `rtGeometryGetMotionBorderMode` `rtGeometrySetMotionRange` `rtGeometrySetMotionSteps`

5.10.2.21 RTResult RTAPI rtGeometrySetMotionRange (
RTgeometry *geometry*,
float *timeBegin*,
float *timeEnd*)

Sets the motion time range for a Geometry node.

Description Sets the inclusive motion time range [*timeBegin*, *timeEnd*] for *geometry*, where *timeBegin* \leq *timeEnd*. The default time range is [0.0, 1.0]. The time range has no effect unless `rtGeometrySetMotionSteps` is called, in which case the time steps uniformly divide the time range. See `rtGeometrySetMotionSteps` for additional requirements on the bounds program.

Parameters

in	<i>geometry</i>	Geometry node handle
out	<i>timeBegin</i>	Beginning time value of range
out	<i>timeEnd</i>	Ending time value of range

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometrySetMotionRange` was introduced in OptiX 5.0.

See also `rtGeometryGetMotionRange` `rtGeometrySetMotionBorderMode` `rtGeometrySetMotionSteps`

5.10.2.22 RTResult RTAPI rtGeometrySetMotionSteps (
RTgeometry *geometry*,
unsigned int *n*)

Specifies the number of motion steps associated with a Geometry.

Description `rtGeometrySetMotionSteps` sets the number of motion steps associated with *geometry*. If the value of *n* is greater than 1, then *geometry* must have an associated bounding box program that takes both a primitive index and a motion index as arguments, and computes an aabb at the motion index. See [rtGeometrySetBoundingBoxProgram](#).

Note that all Geometry has at least one 1 motion step (the default), and Geometry that linearly moves has 2 motion steps.

Parameters

in	<i>geometry</i>	Geometry node handle
in	<i>n</i>	Number of motion steps ≥ 1

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometrySetMotionSteps` was introduced in OptiX 5.0.

See also [rtGeometryGetMotionSteps](#) [rtGeometrySetMotionBorderMode](#) [rtGeometrySetMotionRange](#)

5.10.2.23 RTResult RTAPI `rtGeometrySetPrimitiveCount` (

`RTgeometry geometry,`
`unsigned int primitiveCount)`

Sets the number of primitives.

Description

`rtGeometrySetPrimitiveCount` sets the number of primitives *primitiveCount* in *geometry*.

Parameters

in	<i>geometry</i>	The geometry node for which to set the number of primitives
in	<i>primitiveCount</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`

5.10.2.24 RTResult RTAPI rtGeometrySetPrimitiveIndexOffset (
RTgeometry *geometry*,
unsigned int *indexOffset*)

Sets the primitive index offset.

Description

`rtGeometrySetPrimitiveIndexOffset` sets the primitive index offset *indexOffset* in *geometry*. In the past, a `Geometry` functions object's primitive index range always started at zero (i.e., a `Geometry` with *N* primitives would have a primitive index range of [0,N-1]). The index offset is used to allow `Geometry` functions objects to have primitive index ranges starting at non-zero positions (i.e., 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` functions objects.

Parameters

in	<i>geometry</i>	The geometry node for which to set the primitive index offset
in	<i>indexOffset</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`

5.10.2.25 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`

5.11 GeometryTriangles functions

Functions

- RTresult RTAPI rtGeometryTrianglesCreate (RTcontext context, RTgeometrytriangles *geometrytriangles)
- RTresult RTAPI rtGeometryTrianglesDestroy (RTgeometrytriangles geometrytriangles)
- RTresult RTAPI rtGeometryTrianglesValidate (RTgeometrytriangles geometrytriangles)
- RTresult RTAPI rtGeometryTrianglesGetContext (RTgeometrytriangles geometrytriangles, RTcontext *context)
- RTresult RTAPI rtGeometryTrianglesSetPrimitiveIndexOffset (RTgeometrytriangles geometrytriangles, unsigned int indexOffset)
- RTresult RTAPI rtGeometryTrianglesGetPrimitiveIndexOffset (RTgeometrytriangles geometrytriangles, unsigned int *indexOffset)
- RTresult RTAPI rtGeometryTrianglesSetPreTransformMatrix (RTgeometrytriangles geometrytriangles, int transpose, const float *matrix)
- RTresult RTAPI rtGeometryTrianglesGetPreTransformMatrix (RTgeometrytriangles geometrytriangles, int transpose, float *matrix)
- RTresult RTAPI rtGeometryTrianglesSetPrimitiveCount (RTgeometrytriangles geometrytriangles, unsigned int triangleCount)
- RTresult RTAPI rtGeometryTrianglesGetPrimitiveCount (RTgeometrytriangles geometrytriangles, unsigned int *triangleCount)
- RTresult RTAPI rtGeometryTrianglesSetTriangleIndices (RTgeometrytriangles geometrytriangles, RTbuffer indexBuffer, RTsize indexBufferByteOffset, RTsize triIndicesByteStride, RTformat triIndicesFormat)
- RTresult RTAPI rtGeometryTrianglesSetVertices (RTgeometrytriangles geometrytriangles, unsigned int vertexCount, RTbuffer vertexBuffer, RTsize vertexBufferByteOffset, RTsize vertexByteStride, RTformat positionFormat)
- RTresult RTAPI rtGeometryTrianglesSetMotionVertices (RTgeometrytriangles geometrytriangles, unsigned int vertexCount, RTbuffer vertexBuffer, RTsize vertexBufferByteOffset, RTsize vertexByteStride, RTsize vertexMotionStepByteStride, RTformat positionFormat)
- RTresult RTAPI rtGeometryTrianglesSetMotionVerticesMultiBuffer (RTgeometrytriangles geometrytriangles, unsigned int vertexCount, RTbuffer *vertexBuffers, unsigned int vertexBufferCount, RTsize vertexBufferByteOffset, RTsize vertexByteStride, RTformat positionFormat)
- RTresult RTAPI rtGeometryTrianglesSetMotionSteps (RTgeometrytriangles geometrytriangles, unsigned int motionStepCount)
- RTresult RTAPI rtGeometryTrianglesGetMotionSteps (RTgeometrytriangles geometrytriangles, unsigned int *motionStepCount)
- RTresult RTAPI rtGeometryTrianglesSetMotionRange (RTgeometrytriangles geometrytriangles, float timeBegin, float timeEnd)
- RTresult RTAPI rtGeometryTrianglesGetMotionRange (RTgeometrytriangles geometrytriangles, float *timeBegin, float *timeEnd)
- RTresult RTAPI rtGeometryTrianglesSetMotionBorderMode (RTgeometrytriangles geometrytriangles, RTmotionbordermode beginMode, RTmotionbordermode endMode)
- RTresult RTAPI rtGeometryTrianglesGetMotionBorderMode (RTgeometrytriangles geometrytriangles, RTmotionbordermode *beginMode, RTmotionbordermode *endMode)

- RTresult RTAPI rtGeometryTrianglesSetBuildFlags (RTgeometrytriangles *geometrytriangles*, RTgeometrybuildflags *buildFlags*)
- RTresult RTAPI rtGeometryTrianglesGetMaterialCount (RTgeometrytriangles *geometrytriangles*, unsigned int **numMaterials*)
- RTresult RTAPI rtGeometryTrianglesSetMaterialCount (RTgeometrytriangles *geometrytriangles*, unsigned int *numMaterials*)
- RTresult RTAPI rtGeometryTrianglesSetMaterialIndices (RTgeometrytriangles *geometrytriangles*, RTbuffer *materialIndexBuffer*, RTsize *materialIndexBufferByteOffset*, RTsize *materialIndexByteStride*, RTformat *materialIndexFormat*)
- RTresult RTAPI rtGeometryTrianglesSetFlagsPerMaterial (RTgeometrytriangles *geometrytriangles*, unsigned int *materialIndex*, RTgeometryflags *flags*)
- RTresult RTAPI rtGeometryTrianglesGetFlagsPerMaterial (RTgeometrytriangles *triangles*, unsigned int *materialIndex*, RTgeometryflags **flags*)
- RTresult RTAPI rtGeometryTrianglesSetAttributeProgram (RTgeometrytriangles *geometrytriangles*, RTprogram *program*)
- RTresult RTAPI rtGeometryTrianglesGetAttributeProgram (RTgeometrytriangles *geometrytriangles*, RTprogram **program*)
- RTresult RTAPI rtGeometryTrianglesDeclareVariable (RTgeometrytriangles *geometrytriangles*, const char **name*, RTvariable **v*)
- RTresult RTAPI rtGeometryTrianglesQueryVariable (RTgeometrytriangles *geometrytriangles*, const char **name*, RTvariable **v*)
- RTresult RTAPI rtGeometryTrianglesRemoveVariable (RTgeometrytriangles *geometrytriangles*, RTvariable *v*)
- RTresult RTAPI rtGeometryTrianglesGetVariableCount (RTgeometrytriangles *geometrytriangles*, unsigned int **count*)
- RTresult RTAPI rtGeometryTrianglesGetVariable (RTgeometrytriangles *geometrytriangles*, unsigned int *index*, RTvariable **v*)

5.11.1 Detailed Description

Functions related to an OptiX GeometryTriangles node.

5.11.2 Function Documentation

5.11.2.1 RTresult RTAPI rtGeometryTrianglesCreate (

RTcontext *context*,
RTgeometrytriangles * *geometrytriangles*)

Creates a new GeometryTriangles node.

Description

rtGeometryTrianglesCreate creates a new GeometryTriangles node within a context. *context* specifies the target context, and should be a value returned by *rtContextCreate*. Sets **geometrytriangles* to the handle of a newly created GeometryTriangles node within *context*. Returns

RT_ERROR_INVALID_VALUE if *geometrytriangles* is *NULL*.

Parameters

in	<i>context</i>	Specifies the rendering context of the GeometryTriangles node
out	<i>geometrytriangles</i>	New GeometryTriangles node handle

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE
- RT_ERROR_MEMORY_ALLOCATION_FAILED

History

`rtGeometryTrianglesCreate` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesDestroy`,

5.11.2.2 RTResult RTAPI `rtGeometryTrianglesDeclareVariable` (

```
RTgeometrytriangles geometrytriangles,
const char * name,
RTvariable * v )
```

Declares a geometry variable for a GeometryTriangles object.

Description

`rtGeometryTrianglesDeclareVariable` declares a *variable* attribute of a *geometrytriangles* object with a specified *name*.

Parameters

in	<i>geometrytriangles</i>	A geometry node
in	<i>name</i>	The name of the variable
out	<i>v</i>	A pointer to a handle to the variable

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesDeclareVariable` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesGetVariable`, `rtGeometryTrianglesGetVariableCount`, `rtGeometryTrianglesQueryVariable`, `rtGeometryTrianglesRemoveVariable`

5.11.2.3 RTResult RTAPI rtGeometryTrianglesDestroy (

RTgeometrytriangles *geometrytriangles*)

Destroys a GeometryTriangles node.

Description

`rtGeometryTrianglesDestroy` removes *geometrytriangles* from its context and deletes it. *geometrytriangles* should be a value returned by `rtGeometryTrianglesCreate`. After the call, *geometrytriangles* is no longer a valid handle.

Parameters

in	<i>geometrytriangles</i>	Handle of the GeometryTriangles node to destroy
----	--------------------------	---

Return values

Relevant return values:

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

History

`rtGeometryTrianglesDestroy` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesCreate`, `rtGeometryTrianglesSetPrimitiveCount`, `rtGeometryTrianglesGetPrimitiveCount`

5.11.2.4 RTResult RTAPI rtGeometryTrianglesGetAttributeProgram (

RTgeometrytriangles *geometrytriangles*,
RTprogram * *program*)

Gets the attribute program of a GeometryTriangles object.

Description

`rtGeometryTrianglesGetAttributeProgram` gets the attribute *program* of a given *geometrytriangles* object. If no program has been set, 0 is returned.

Parameters

in	<i>geometrytriangles</i>	The geometrytriangles node for which to set the attribute program
out	<i>program</i>	A pointer to a handle to the attribute program

Return values

Relevant return values:

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

- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesGetAttributeProgram` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesDeclareVariable`, `rtGeometryTrianglesSetAttributeProgram`, `rtProgramCreateFromPTXFile`, `rtProgramCreateFromPTXString`

5.11.2.5 RTResult RTAPI `rtGeometryTrianglesGetContext` (

`RTgeometrytriangles geometrytriangles,`
`RTcontext * context)`

Returns the context associated with a `GeometryTriangles` node.

Description

`rtGeometryTrianglesGetContext` queries a `GeometryTriangles` node for its associated context. `geometrytriangles` specifies the `GeometryTriangles` node to query, and should be a value returned by `rtGeometryTrianglesCreate`. Sets `*context` to the context associated with `geometrytriangles`.

Parameters

in	<code>geometrytriangles</code>	Specifies the <code>GeometryTriangles</code> to query
out	<code>context</code>	The context associated with <code>geometrytriangles</code>

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE
- RT_ERROR_MEMORY_ALLOCATION_FAILED

History

`rtGeometryTrianglesGetContext` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesCreate`

5.11.2.6 RTResult RTAPI `rtGeometryTrianglesGetFlagsPerMaterial` (

`RTgeometrytriangles triangles,`
`unsigned int materialIndex,`
`RTgeometryflags * flags)`

Gets geometry flags for triangles.

Description

See `rtGeometryTrianglesSetFlagsPerMaterial` for details.

Parameters

in	<i>triangles</i>	The triangles handle
in	<i>materialIndex</i>	The index of the material for which to retrieve the flags
out	<i>flags</i>	Flags for the given geometry group

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesGetFlagsPerMaterial` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetFlagsPerMaterial`, `rtGeometryTrianglesSetMaterialIndices` `rtTrace`

5.11.2.7 RTResult RTAPI `rtGeometryTrianglesGetMaterialCount` (

`RTgeometrytriangles geometrytriangles,`
`unsigned int * numMaterials)`

Sets the number of materials used for the `GeometryTriangles`.

Description `rtGeometryTrianglesGetMaterialCount` returns the number of materials that are used with `geometrytriangles`. As default there is one material slot.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node handle
out	<i>numMaterials</i>	Number of materials used with this GeometryTriangles node

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesGetMaterialCount` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetMaterialCount`

5.11.2.8 RTResult RTAPI `rtGeometryTrianglesGetMotionBorderMode` (

`RTgeometrytriangles geometrytriangles,`
`RTmotionbordermode * beginMode,`

RTmotionbordermode * *endMode*)

Returns the motion border modes of a GeometryTriangles node.

Description `rtGeometryTrianglesGetMotionBorderMode` returns the motion border modes for the time range associated with *geometrytriangles*.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node handle
out	<i>beginMode</i>	Motion border mode at motion range begin
out	<i>endMode</i>	Motion border mode at motion range end

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

History

`rtGeometryTrianglesGetMotionBorderMode` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetMotionBorderMode` `rtGeometryTrianglesGetMotionRange` `rtGeometryTrianglesGetMotionSteps`

5.11.2.9 RTResult RTAPI rtGeometryTrianglesGetMotionRange (

```
RTgeometrytriangles geometrytriangles,
float * timeBegin,
float * timeEnd )
```

Returns the motion time range associated with a GeometryTriangles node.

Description `rtGeometryTrianglesGetMotionRange` returns the motion time range associated with *geometrytriangles* from a previous call to `rtGeometryTrianglesSetMotionRange`, or the default values of [0.0, 1.0].

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node handle
out	<i>timeBegin</i>	Beginning time value of range
out	<i>timeEnd</i>	Ending time value of range

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`

- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesGetMotionRange` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetMotionRange` `rtGeometryTrianglesGetMotionBorderMode`
`rtGeometryTrianglesGetMotionSteps`

5.11.2.10 RTResult RTAPI `rtGeometryTrianglesGetMotionSteps` (

`RTgeometrytriangles geometrytriangles,`
`unsigned int * motionStepCount)`

Returns the number of motion steps associated with a `GeometryTriangles` node.

Description `rtGeometryTrianglesGetMotionSteps` returns in `motionStepCount` the number of motion steps associated with `geometrytriangles`. Note that the default value is 1, not 0, for geometry without motion.

Parameters

in	<code>geometrytriangles</code>	GeometryTriangles node handle
out	<code>motionStepCount</code>	Number of motion steps <code>motionStepCount >= 1</code>

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesGetMotionSteps` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetMotionSteps` `rtGeometryTrianglesGetMotionBorderMode`
`rtGeometryTrianglesGetMotionRange`

5.11.2.11 RTResult RTAPI `rtGeometryTrianglesGetPreTransformMatrix` (

`RTgeometrytriangles geometrytriangles,`
`int transpose,`
`float * matrix)`

Gets a pre-transform matrix.

Description

`rtGeometryTrianglesGetPreTransformMatrix` returns a previously set 3x4 matrix or the 'identity' matrix (with ones in the main diagonal of the 3x3 submatrix) if no matrix is set.

Parameters

in	<i>geometrytriangles</i>	Geometry node to query from the number of primitives
in	<i>transpose</i>	Set to true if the output matrix is expected to be column-major rather than row-major
out	<i>matrix</i>	The 3x4 matrix that is used to transform the vertices

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesGetPreTransformMatrix` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetPreTransformMatrix`

5.11.2.12 RTResult RTAPI rtGeometryTrianglesGetPrimitiveCount (
RTgeometrytriangles *geometrytriangles*,
unsigned int * *triangleCount*)

Returns the number of triangles.

Description

`rtGeometryTrianglesGetPrimitiveCount` returns the number of set triangles for *geometrytriangles*. The number of primitives can be set with `rtGeometryTrianglesSetPrimitiveCount`.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node to query from the number of primitives
out	<i>triangleCount</i>	Number of triangles

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesGetPrimitiveCount` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetPrimitiveCount` `rtGeometryGetPrimitiveCount`

5.11.2.13 RTResult RTAPI rtGeometryTrianglesGetPrimitiveIndexOffset (

```
RTgeometrytriangles geometrytriangles,
unsigned int * indexOffset )
```

Returns the current primitive index offset.

Description

`rtGeometryTrianglesGetPrimitiveIndexOffset` returns for *geometrytriangles* the primitive index offset. The primitive index offset can be set with `rtGeometryTrianglesSetPrimitiveIndexOffset`.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node to query for the primitive index offset
out	<i>indexOffset</i>	Primitive index offset

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesGetPrimitiveIndexOffset` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetPrimitiveIndexOffset`

5.11.2.14 RTResult RTAPI `rtGeometryTrianglesGetVariable` (

```
RTgeometrytriangles geometrytriangles,
unsigned int index,
RTvariable * v )
```

Get a variable attached to a GeometryTriangles object at a specified index.

Description

`rtGeometryTrianglesGetVariable` returns the variable attached at a given index to the specified GeometryTriangles object.

Parameters

in	<i>geometrytriangles</i>	A geometry node
in	<i>index</i>	The index of the variable
out	<i>v</i>	A pointer to a variable handle

Return values

Relevant return values:

- RT_SUCCESS

- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesGetVariable` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesDeclareVariable`, `rtGeometryTrianglesGetVariableCount`, `rtGeometryTrianglesQueryVariable`, `rtGeometryTrianglesRemoveVariable`

5.11.2.15 RTResult RTAPI rtGeometryTrianglesGetVariableCount (
RTgeometrytriangles *geometrytriangles*,
unsigned int * *count*)

Get the number of variables attached to a `GeometryTriangles` object.

Description

`rtGeometryTrianglesGetVariableCount` returns a *count* of the number of variables attached to a *geometrytriangles* object.

Parameters

in	<i>geometrytriangles</i>	A <code>geometrytriangles</code> node
out	<i>v</i>	A pointer to an <code>unsigned int</code>

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT

History

`rtGeometryTrianglesGetVariableCount` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesDeclareVariable`, `rtGeometryTrianglesGetVariable`, `rtGeometryTrianglesQueryVariable`, `rtGeometryTrianglesRemoveVariable`

5.11.2.16 RTResult RTAPI rtGeometryTrianglesQueryVariable (
RTgeometrytriangles *geometrytriangles*,
const char * *name*,
RTvariable * *v*)

Queries a variable attached to a `GeometryTriangles` object.

Description

`rtGeometryTrianglesQueryVariable` gets a variable with a given *name* from a *geometrytriangles* object.

Parameters

in	<i>geometrytriangles</i>	A <code>geometrytriangles</code> object
----	--------------------------	---

Parameters

in	<i>name</i>	Thee name of the variable
out	<i>v</i>	A pointer to a handle to the variable

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesQueryVariable` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesGetVariable`, `rtGeometryTrianglesGetVariableCount`, `rtGeometryTrianglesQueryVariable`, `rtGeometryTrianglesRemoveVariable`

5.11.2.17 RTResult RTAPI rtGeometryTrianglesRemoveVariable (
RTgeometrytriangles *geometrytriangles*,
RTvariable *v*)

Removes a variable from `GeometryTriangles` object.

Description

`rtGeometryTrianglesRemoveVariable` removes a variable from a `geometrytriangles` object.

Parameters

in	<i>geometrytriangles</i>	A <code>geometrytriangles</code> object
in	<i>v</i>	A pointer to a handle to the variable

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesRemoveVariable` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesDeclareVariable`, `rtGeometryTrianglesGetVariable`, `rtGeometryTrianglesGetVariableCount`, `rtGeometryTrianglesQueryVariable`

5.11.2.18 RTResult RTAPI rtGeometryTrianglesSetAttributeProgram (
RTgeometrytriangles *geometrytriangles*,

RTprogram *program*)

Sets the attribute program on a GeometryTriangles object.

Description

`rtGeometryTrianglesSetAttributeProgram` sets for *geometrytriangles* the *program* that performs attribute computation. RTprograms can be either generated with `rtProgramCreateFromPTXFile` or `rtProgramCreateFromPTXString`. An attribute program is optional. If no attribute program is specified, a default attribute program will be provided. Attributes are computed after intersection and before any hit or closest hit programs that require those attributes. No assumptions about the precise invocation time should be made.

The default attribute program will provide the following attributes: float2 barycentrics; unsigned int instanceid;

Names are case sensitive and types must match. To use the attributes, declare the following
`rtDeclareVariable(float2, barycentrics, attribute barycentrics,);`
`rtDeclareVariable(unsigned int, instanceid, attribute instanceid,);`

If you provide an attribute program, the following device side functions will be available. float2
`rtGetTriangleBarycentrics();` unsigned int `rtGetInstanceId();`

These device functions are only available in attribute programs.

Parameters

in	<i>geometrytriangles</i>	The geometrytriangles node for which to set the attribute program
in	<i>program</i>	A handle to the attribute program

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

History

`rtGeometryTrianglesSetAttributeProgram` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesGetAttributeProgram`, `rtProgramCreateFromPTXFile`, `rtProgramCreateFromPTXString`, `rtGetTriangleBarycentrics`, `rtGetInstanceId`

5.11.2.19 RTResult RTAPI rtGeometryTrianglesSetBuildFlags (

RTgeometrytriangles *geometrytriangles*,
RTgeometrybuildflags *buildFlags*)

Sets flags that influence the behavior of traversal.

Description `rtGeometryTrianglesSetBuildFlags` can be used to set object-specific flags that affect the acceleration-structure-build behavior. If parameter *buildFlags* contains the `RT_GEOMETRY_BUILD_FLAG_RELEASE_BUFFERS` flag, all buffers (including the vertex, index, and materialIndex buffer) holding information that is evaluated at acceleration-structure-build time will

be released after the build. OptiX does not take ownership over the buffers, but simply frees the corresponding device memory. Sharing buffers with other GeometryTriangles nodes is possible if all of them are built within one OptiX launch. Note that it is the users responsibility that the buffers hold data for the next acceleration structure build if the acceleration structure is marked dirty. E.g., if the flag is set, an OptiX launch will cause the acceleration structure build and release the memory afterwards. If the acceleration structure is marked dirty before the next launch (e.g., due to refitting), the user needs to map the buffers before the launch to fill them with data. Further, there are certain configurations with motion when the buffers cannot be released in which case the flag is ignored and the data is not freed. The buffers can only be released if all GeometryTriangles belonging to a GeometryGroup have the same number of motion steps and equal motion begin / end times.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node handle
in	<i>buildFlags</i>	The flags to set

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetBuildFlags` was introduced in OptiX 6.0.

See also [rtGeometryTrianglesSetBuildFlags](#)

5.11.2.20 RTResult RTAPI `rtGeometryTrianglesSetFlagsPerMaterial` (

`RTgeometrytriangles geometrytriangles,`
`unsigned int materialIndex,`
`RTgeometryflags flags)`

Sets geometry-specific flags that influence the behavior of traversal.

Description `rtGeometryTrianglesSetFlagsPerMaterial` can be used to set geometry-specific flags that will eventually change the behavior of traversal when intersecting the geometry. Note that the flags are evaluated at acceleration-structure-build time. An acceleration must be marked dirty for changes to the flags to take effect. Setting the flags `RT_GEOMETRY_FLAG_NO_SPLITTING` and/or `RT_GEOMETRY_FLAG_DISABLE_ANYHIT` should be dependent on the material that is used for the intersection. Therefore, the flags are set per material slot (with the actual material binding begin set at the `GeomtryInstance`). If the geometry is instanced and different instances apply different materials to the geometry, the per-material geometry-specific flags need to apply to the materials of all instances. Example with two instances with each having two materials, node graph: `G /\ \ T0 T1 || GG0-A-GG1 | M0-GI0 GI1-M2 /\ \ M1 GT M3` with: G-Group, GG-GeometryGroup, T-Transform, A-Acceleration, GI-GeometryInstance, M-Material, GT-GeometryTriangles `RT_GEOMETRY_FLAG_NO_SPLITTING` needs to be set for material index 0, if M0 or M2 require it. `RT_GEOMETRY_FLAG_DISABLE_ANYHIT` should be set for material index 0, if M0 and M2 allow it. `RT_GEOMETRY_FLAG_NO_SPLITTING`

needs to be set for material index 1, if M1 or M3 require it. RT_GEOMETRY_FLAG_DISABLE_ANYHIT should be set for material index 1, if M1 and M3 allow it.

Setting RT_GEOMETRY_FLAG_NO_SPLITTING prevents splitting the primitive during the bvh build. Splitting is done to increase performance, but as a side-effect may result in multiple executions of the any hit program for a single intersection. To avoid further side effects (e.g., multiple accumulations of a value) that may result of a multiple execution, RT_GEOMETRY_FLAG_NO_SPLITTING needs to be set. RT_GEOMETRY_FLAG_DISABLE_ANYHIT is an optimization due to which the execution of the any hit program is skipped. If possible, the flag should be set. Note that even if no any hit program is set on a material, this flag needs to be set to skip the any hit program. This requirement is because the information whether or not to skip the any hit program needs to be available at bvh build time (while materials can change afterwards without a bvh rebuild). Note that the final decision whether or not to execute the any hit program at run time also depends on the flags set on the ray as well as the geometry group that this geometry is part of.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node handle
in	<i>materialIndex</i>	The material index for which to set the flags
in	<i>flags</i>	The flags to set.

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetFlagsPerMaterial` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetMaterialCount` `rtGeometryTrianglesSetMaterialIndices`
`rtGeometryTrianglesSetBuildFlags`

5.11.2.21 RTResult RTAPI rtGeometryTrianglesSetMaterialCount (
RTgeometrytriangles *geometrytriangles*,
unsigned int *numMaterials*)

Sets the number of materials used for the `GeometryTriangles`.

Description `rtGeometryTrianglesSetMaterialCount` sets the number of materials that are used with `geometrytriangles`. As default, there is one material slot. This number must be equal to the number of materials that is set at the `GeometryInstance` where `geometrytriangles` is attached to. Multi-material support for `GeometryTriangles` is limited to a fixed partition of the geometry into sets of triangles. Each triangle set maps to one material slot (within range [0;`numMaterials`]). The mapping is set via `rtGeometryTrianglesSetMaterialIndices`. The actual materials are set at the `GeometryInstance`. The geometry can be instanced when attached to multiple `GeometryInstances`. In that case, the materials attached to each `GeometryInstance` can differ (effectively causing different materials per instance of the geometry). `numMaterials` must be ≥ 1 and $\leq 2^{16}$.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node handle
in	<i>numMaterials</i>	Number of materials used with this geometry

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetMaterialCount` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesGetMaterialCount` `rtGeometryTrianglesSetMaterialIndices`
`rtGeometryTrianglesSetFlagsPerMaterial`

5.11.2.22 RTResult RTAPI `rtGeometryTrianglesSetMaterialIndices` (
RTgeometrytriangles *geometrytriangles*,
RTbuffer *materialIndexBuffer*,
RTsize *materialIndexBufferByteOffset*,
RTsize *materialIndexByteStride*,
RTformat *materialIndexFormat*)

Sets the index buffer of indexed triangles.

Description

`rtGeometryTrianglesSetMaterialIndices` set the material slot per triangle of *geometrytriangles*. Hence, buffer *materialIndexBuffer* must hold triangleCount entries. Every material index must be in range see 0; *numMaterials*-1. Parameter *materialIndexBufferByteOffset* can be used to specify a byte offset to the first index in buffer *materialIndexBuffer*. Parameter *materialIndexByteStride* sets the stride in bytes between indices. Parameter *materialIndexFormat* must be one of the following:

`RT_FORMAT_UNSIGNED_INT`, `RT_FORMAT_UNSIGNED_SHORT`,
`RT_FORMAT_UNSIGNED_BYTE`. The buffer is only used if the number of materials as set via `rtGeometryTrianglesSetMaterialCount` is larger than one.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node to query for the primitive index offset
in	<i>materialIndexBuffer</i>	Buffer that holds the indices into the vertex buffer of the triangles
in	<i>materialIndexBufferByteOffset</i>	Offset to first index in buffer <i>indexBuffer</i>
in	<i>materialIndexByteStride</i>	Stride in bytes between triplets of indices
in	<i>materialIndexFormat</i>	Format of the triplet of indices to index the vertices of a triangle

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetMaterialIndices` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetMaterialCount` `rtGeometryTrianglesSetFlagsPerMaterial`

5.11.2.23 RTResult RTAPI `rtGeometryTrianglesSetMotionBorderMode` (

RTgeometrytriangles *geometrytriangles*,
RTmotionbordermode *beginMode*,
RTmotionbordermode *endMode*)

Sets the motion border modes of a `GeometryTriangles` node.

Description `rtGeometryTrianglesSetMotionBorderMode` sets the behavior of *geometrytriangles* outside its motion time range. Options are `RT_MOTIONBORDERMODE_CLAMP` or `RT_MOTIONBORDERMODE_VANISH`. See `rtTransformSetMotionBorderMode` for details.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node handle
in	<i>beginMode</i>	Motion border mode at motion range begin
in	<i>endMode</i>	Motion border mode at motion range end

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetMotionBorderMode` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesGetMotionBorderMode` `rtGeometryTrianglesSetMotionRange` `rtGeometryTrianglesGetMotionSteps`

5.11.2.24 RTResult RTAPI `rtGeometryTrianglesSetMotionRange` (

RTgeometrytriangles *geometrytriangles*,
float *timeBegin*,
float *timeEnd*)

Sets the motion time range for a `GeometryTriangles` node.

Description Sets the inclusive motion time range [*timeBegin*, *timeEnd*] for *geometrytriangles*, where *timeBegin* <= *timeEnd*. The default time range is [0.0, 1.0]. The time range has no effect unless

`rtGeometryTrianglesSetMotionVertices` or `rtGeometryTrianglesSetMotionVerticesMultiBuffer` with `motionStepCount > 1` is called, in which case the time steps uniformly divide the time range.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node handle
out	<i>timeBegin</i>	Beginning time value of range
out	<i>timeEnd</i>	Ending time value of range

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetMotionRange` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesGetMotionRange` `rtGeometryTrianglesSetMotionBorderMode`
`rtGeometryTrianglesGetMotionSteps`

5.11.2.25 RTResult RTAPI `rtGeometryTrianglesSetMotionSteps` (
RTgeometrytriangles *geometrytriangles*,
unsigned int *motionStepCount*)

Sets the number of motion steps associated with a `GeometryTriangles` node.

Description `rtGeometryTrianglesSetMotionSteps` sets the number of motion steps as specified in `motionStepCount` associated with `geometrytriangles`. Note that the default value is 1, not 0, for geometry without motion.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node handle
in	<i>motionStepCount</i>	Number of motion steps, <code>motionStepCount >= 1</code>

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesGetMotionSteps` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetMotionVertices` `rtGeometryTrianglesSetMotionVerticesMultiBuffer`

```
rtGeometryTrianglesGetMotionSteps rtGeometryTrianglesSetMotionBorderMode
rtGeometryTrianglesSetMotionRange
```

5.11.2.26 RTResult RTAPI rtGeometryTrianglesSetMotionVertices (

```
RTgeometrytriangles geometrytriangles,
unsigned int vertexCount,
RTbuffer vertexBuffer,
RTsize vertexBufferByteOffset,
RTsize vertexByteStride,
RTsize vertexMotionStepByteStride,
RTformat positionFormat )
```

Sets the vertex buffer of motion triangles.

Description

`rtGeometryTrianglesSetMotionVertices` interprets the buffer `vertexBuffer` as the vertices of triangles of the GeometryTriangles `geometrytriangles`. The number of triangles for one motion step is set as `vertexCount`. Similar to its non-motion counterpart, `vertexCount` must be 3 times `triangleCount` if no index buffer is set. The total number of vertices stored in `vertexBuffer` is `vertexCount` times `motionStepCount` (see `rtGeometryTrianglesSetMotionSteps`). Triangles are linearly interpolated between motion steps. Parameter `vertexBufferByteOffset` can be used to specify a byte offset to the position of the first vertex of the first motion step in buffer `vertexBuffer`. Parameter `vertexByteStride` sets the stride in bytes between vertices within a motion step. Parameter `vertexMotionStepByteStride` sets the stride in bytes between motion steps for a single vertex. The stride parameters allow for two types of layouts of the motion data: a) serialized: `vertexByteStride` = `sizeof(Vertex)`, `vertexMotionStepByteStride` = `vertexCount * vertexByteStride` b) interleaved: `motion_step_byte_stride` = `sizeof(Vertex)`, `vertexByteStride` = `sizeof(Vertex)` * `motion_steps` Vertex N at time step i is at: `vertexBuffer[N * vertexByteStride + i * vertexMotionStepByteStride + vertexBufferByteOffset]` Parameter `positionFormat` must be one of the following: `RT_FORMAT_FLOAT3`, `RT_FORMAT_HALF3`, `RT_FORMAT_FLOAT2`, `RT_FORMAT_HALF2`. In case of formats `RT_FORMAT_FLOAT2` or `RT_FORMAT_HALF2` the third component is assumed to be zero, which can be useful for planar geometry. Calling this function overrides any previous call to anyone of the `set(Motion)Vertices` functions.

Parameters

in	<code>geometrytriangles</code>	GeometryTriangles node to query for the primitive index offset
in	<code>vertexCount</code>	Number of vertices for one motion step
in	<code>vertexBuffer</code>	Buffer that holds the vertices of the triangles for all motion steps
in	<code>vertexBufferByteOffset</code>	Offset in bytes to the first vertex of the first motion step in buffer <code>vertexBuffer</code>
in	<code>vertexByteStride</code>	Stride in bytes between vertices, belonging to the same motion step
in	<code>vertexMotionStepByteStride</code>	Stride in bytes between vertices of the same triangle, but neighboring motion step
in	<code>positionFormat</code>	Format of the position attribute of a vertex

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetMotionVertices` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetVertices` `rtGeometryTrianglesSetMotionVerticesMultiBuffer`

5.11.2.27 RTResult RTAPI `rtGeometryTrianglesSetMotionVerticesMultiBuffer` (

```
RTgeometrytriangles geometrytriangles,
unsigned int vertexCount,
RTbuffer * vertexBuffers,
unsigned int vertexBufferCount,
RTsize vertexBufferByteOffset,
RTsize vertexByteStride,
RTformat positionFormat )
```

Sets the vertex buffer of motion triangles.

Description

`rtGeometryTrianglesSetMotionVerticesMultiBuffer` can be used instead of `rtGeometryTrianglesSetMotionVertices` if the vertices for the different motion steps are stored in separate buffers. Parameter *vertexBuffers* must point to an array of buffers of minimal size *motionStepCount* (see `rtGeometryTrianglesSetMotionSteps`). All buffers must, however, share the same byte offset as well as vertex stride and position format. Calling this function overrides any previous call to any of the `set(Motion)Vertices` functions.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node to query for the primitive index offset
in	<i>vertexCount</i>	Number of vertices for one motion step
in	<i>vertexBuffers</i>	Buffers that hold the vertices of the triangles per motion step
in	<i>vertexBufferCount</i>	Number of buffers passed, must match the number of motion steps before a launch call
in	<i>vertexBufferByteOffset</i>	Offset in bytes to the first vertex in every buffer <i>vertexBuffers</i>
in	<i>vertexByteStride</i>	Stride in bytes between vertices, belonging to the same motion step
in	<i>positionFormat</i>	Format of the position attribute of a vertex

Return values

Relevant return values:

- RT_SUCCESS

- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetMotionVertices` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetVertices` `rtGeometryTrianglesSetMotionVertices`

5.11.2.28 RTResult RTAPI `rtGeometryTrianglesSetPreTransformMatrix` (

```
RTgeometrytriangles geometrytriangles,
int transpose,
const float * matrix )
```

Sets a pre-transform matrix.

Description

`rtGeometryTrianglesSetPreTransformMatrix` can be used to bake a transformation for a mesh. Vertices of triangles are multiplied by the user-specified 3x4 matrix before the acceleration build. Note that the input triangle data stays untouched (set via `rtGeometryTrianglesSetVertices`). Triangle intersection uses transformed triangles. The 3x4 matrix is expected to be in a row-major data layout, use the transpose option if *matrix* is in a column-major data layout. Use `rtGeometryTrianglesSetPreTransformMatrix(geometrytriangles, false, 0);` to unset a previously set matrix.

Parameters

in	<i>geometrytriangles</i>	Geometry node to query from the number of primitives
in	<i>transpose</i>	If the input matrix is column-major and needs to be transposed before usage
in	<i>matrix</i>	The 3x4 matrix that is used to transform the vertices

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetPreTransformMatrix` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesGetPreTransformMatrix`

5.11.2.29 RTResult RTAPI `rtGeometryTrianglesSetPrimitiveCount` (

```
RTgeometrytriangles geometrytriangles,
unsigned int triangleCount )
```

Sets the number of triangles.

Description

`rtGeometryTrianglesSetPrimitiveCount` sets the number of triangles *triangleCount* in *geometrytriangles*. A triangle geometry is either a triangle soup for which every three vertices stored in the vertex buffer form a triangle, or indexed triangles are used for which three indices reference different vertices. In the latter case, an index buffer must be set (`rtGeometryTrianglesSetTriangleIndices`). The vertices of the triangles are specified via one of the `SetVertices` functions.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node for which to set the number of triangles
in	<i>triangleCount</i>	Number of triangles

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetPrimitiveCount` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesGetPrimitiveCount` `rtGeometrySetPrimitiveCount`

5.11.2.30 RTresult RTAPI `rtGeometryTrianglesSetPrimitiveIndexOffset` (
RTgeometrytriangles *geometrytriangles*,
unsigned int *indexOffset*)

Sets the primitive index offset.

Description

`rtGeometryTrianglesSetPrimitiveIndexOffset` sets the primitive index offset *indexOffset* in *geometrytriangles*. With an offset of zero, a `GeometryTriangles` with *N* triangles has a primitive index range of [0,N-1]. The index offset is used to allow `GeometryTriangles` objects to have primitive index ranges starting at non-zero positions (i.e., a `GeometryTriangles` with *N* triangles and an index offset of *M* has a primitive index range of [*M,M+N-1*]). Note that this offset only affects the primitive index that is reported in case of an intersection and does not affect the input data that is specified via `rtGeometryTrianglesSetVertices` or `rtGeometryTrianglesSetTriangleIndices`. This feature enables the packing of multiple Geometries or `GeometryTriangles` into a single buffer. While the same effect could be reached via a user variable, it is recommended to specify the offset via `rtGeometryTrianglesSetPrimitiveIndexOffset`.

Parameters

in	<i>geometrytriangles</i>	The <code>GeometryTriangles</code> node for which to set the primitive index offset
in	<i>indexOffset</i>	The primitive index offset

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetPrimitiveIndexOffset` was introduced in OptiX 6.0.

See also `rtGeometrySetPrimitiveIndexOffset` `rtGeometryTrianglesGetPrimitiveIndexOffset`

5.11.2.31 RTResult RTAPI `rtGeometryTrianglesSetTriangleIndices` (

`RTgeometrytriangles geometrytriangles,`
`RTbuffer indexBuffer,`
`RTsize indexBufferByteOffset,`
`RTsize trilIndicesByteStride,`
`RTformat trilIndicesFormat)`

Sets the index buffer of indexed triangles.

Description

`rtGeometryTrianglesSetTriangleIndices` is used to set the index buffer for indexed triangles. Triplets of indices from buffer *indexBuffer* index vertices to form triangles. If the buffer is set, it is assumed that the geometry is given as indexed triangles. If the index buffer is not set, it is assumed that the geometry is given as a triangle soup. A previously set index buffer can be unset by passing NULL as *indexBuffer* parameter, e.g., `rtGeometryTrianglesSetTriangleIndices(geometrytriangles, NULL, 0, 0, RT_FORMAT_UNSIGNED_INT3);` Buffer *indexBuffer* is expected to hold 3 times *triangleCount* indices (see `rtGeometryTrianglesSetPrimitiveCount`). Parameter *indexBufferByteOffset* can be used to specify a byte offset to the first index in buffer *indexBuffer*. Parameter *trilIndicesByteStride* sets the stride in bytes between triplets of indices. There mustn't be any spacing between indices within a triplet, spacing is only supported between triplets. Parameter *trilIndicesFormat* must be one of the following: `RT_FORMAT_UNSIGNED_INT3`, `RT_FORMAT_UNSIGNED_SHORT3`.

Parameters

in	<i>geometrytriangles</i>	GeometryTriangles node to query for the primitive index offset
in	<i>indexBuffer</i>	Buffer that holds the indices into the vertex buffer of the triangles
in	<i>indexBufferByteOffset</i>	Offset in bytes to the first index in buffer <i>indexBuffer</i>
in	<i>trilIndicesByteStride</i>	Stride in bytes between triplets of indices
in	<i>trilIndicesFormat</i>	Format of the triplet of indices to index the vertices of a triangle

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtGeometryTrianglesSetTriangleIndices` was introduced in OptiX 6.0.

See also `rtGeometryTrianglesSetVertices`

5.11.2.32 RTResult RTAPI `rtGeometryTrianglesSetVertices` (

`RTgeometrytriangles geometrytriangles,`

`unsigned int vertexCount,`

`RTbuffer vertexBuffer,`

`RTsize vertexBufferByteOffset,`

`RTsize vertexByteStride,`

`RTformat positionFormat)`

Sets the vertex buffer of a triangle soup.

Description

`rtGeometryTrianglesSetVertices` interprets the buffer `vertexBuffer` as the vertices of triangles of the GeometryTriangles `geometrytriangles`. The number of vertices is set as `vertexCount`. If an index buffer is set, it is assumed that the geometry is given as indexed triangles. If the index buffer is not set, it is assumed that the geometry is given as a triangle soup and `vertexCount` must be 3 times `triangleCount` (see `rtGeometryTrianglesSetPrimitiveCount`). Buffer `vertexBuffer` is expected to hold `vertexCount` vertices. Parameter `vertexBufferByteOffset` can be used to specify a byte offset to the position of the first vertex in buffer `vertexBuffer`. Parameter `vertexByteStride` sets the stride in bytes between vertices. Parameter `positionFormat` must be one of the following: `RT_FORMAT_FLOAT3`, `RT_FORMAT_HALF3`, `RT_FORMAT_FLOAT2`, `RT_FORMAT_HALF2`. In case of formats `RT_FORMAT_FLOAT2` or `RT_FORMAT_HALF2` the third component is assumed to be zero, which can be useful for planar geometry. Calling this function overrides any previous call to anyone of the `set(Motion)Vertices` functions.

Parameters

in	<code>geometrytriangles</code>	GeometryTriangles node to query for the primitive index offset
in	<code>vertexCount</code>	Number of vertices of the geometry
in	<code>vertexBuffer</code>	Buffer that holds the vertices of the triangles
in	<code>vertexByteStride</code>	Stride in bytes between vertices
in	<code>vertexBufferByteOffset</code>	Offset in bytes to the first vertex in buffer <code>vertexBuffer</code>
in	<code>positionFormat</code>	Format of the position attribute of a vertex

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_CONTEXT`
- `RT_ERROR_INVALID_VALUE`

History

`rtGeometryTrianglesSetVertices` was introduced in OptiX 6.0.

See also [rtGeometryTrianglesSetTriangleIndices](#) [rtGeometryTrianglesSetMotionVertices](#)

5.11.2.33 RTResult RTAPI **rtGeometryTrianglesValidate** (

RTgeometrytriangles *geometrytriangles*)

Validates the GeometryTriangles nodes integrity.

Description

`rtGeometryTrianglesValidate` checks *geometrytriangles* for completeness. If *geometrytriangles* or any of the objects attached to *geometrytriangles* are not valid, returns [RT_ERROR_INVALID_VALUE](#).

Parameters

in	<i>geometrytriangles</i>	The GeometryTriangles 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

`rtGeometryTrianglesValidate` was introduced in OptiX 6.0.

See also [rtContextValidate](#)

5.12 Material functions

Functions

- 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 rayTypeIndex, RTprogram program)
- RTresult RTAPI rtMaterialGetClosestHitProgram (RTmaterial material, unsigned int rayTypeIndex, RTprogram *program)
- RTresult RTAPI rtMaterialSetAnyHitProgram (RTmaterial material, unsigned int rayTypeIndex, RTprogram program)
- RTresult RTAPI rtMaterialGetAnyHitProgram (RTmaterial material, unsigned int rayTypeIndex, 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)

5.12.1 Detailed Description

Functions related to an OptiX Material.

5.12.2 Function Documentation

5.12.2.1 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`

5.12.2.2 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`

5.12.2.3 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`

5.12.2.4 RTResult RTAPI rtMaterialGetAnyHitProgram (

RTmaterial material,
unsigned int rayTypeIndex,
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`. *rayTypeIndex* 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*, *rayTypeIndex*). 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>rayTypeIndex</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`

5.12.2.5 RTresult RTAPI `rtMaterialGetClosestHitProgram` (

```
RTmaterial material,
unsigned int rayTypeIndex,
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`. *rayTypeIndex* 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*, *rayTypeIndex*). 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>rayTypeIndex</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`

5.12.2.6 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](#)

5.12.2.7 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`

5.12.2.8 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`

5.12.2.9 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*

5.12.2.10 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`

5.12.2.11 RTResult RTAPI `rtMaterialSetAnyHitProgram` (
RTmaterial *material*,
unsigned int *rayTypeIndex*,
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`. *rayTypeIndex* 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*, *rayTypeIndex*) 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>rayTypeIndex</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`

5.12.2.12 RTResult RTAPI `rtMaterialSetClosestHitProgram` (
RTmaterial *material*,
unsigned int *rayTypeIndex*,

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`. *rayTypeIndex* 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*, *rayTypeIndex*) 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>rayTypeIndex</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`

5.12.2.13 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](#)

5.13 Program functions

Functions

- RTresult RTAPI rtProgramCreateFromPTXString (RTcontext context, const char *ptx, const char *programName, RTprogram *program)
- RTresult RTAPI rtProgramCreateFromPTXStrings (RTcontext context, unsigned int n, const char **ptxStrings, const char *programName, RTprogram *program)
- RTresult RTAPI rtProgramCreateFromPTXFile (RTcontext context, const char *filename, const char *programName, RTprogram *program)
- RTresult RTAPI rtProgramCreateFromPTXFiles (RTcontext context, unsigned int n, const char **filenames, const char *programName, 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 *programId)
- RTresult RTAPI rtProgramCallsiteSetPotentialCallees (RTprogram program, const char *name, const int *ids, int numIds)
- RTresult RTAPI rtContextGetProgramFromId (RTcontext context, int programId, RTprogram *program)

5.13.1 Detailed Description

Functions related to an OptiX program.

5.13.2 Function Documentation

5.13.2.1 RTresult RTAPI rtContextGetProgramFromId (

```
RTcontext context,
int programId,
RTprogram * program )
```

Gets an RTprogram corresponding to the program id.

Description

rtContextGetProgramFromId returns a handle to the program in **program* corresponding to the *programId* supplied. If *programId* is not a valid program handle, **program* is set to *NULL*. Returns *RT_ERROR_INVALID_VALUE* if *context* is invalid or *programId* is not a valid program handle.

Parameters

in	<i>context</i>	The context the program should be originated from
in	<i>programId</i>	The ID of the program to query
out	<i>program</i>	The return handle for the program object corresponding to the programId

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtContextGetProgramFromId` was introduced in OptiX 3.6.

See also [rtProgramGetId](#)

5.13.2.2 RTresult RTAPI `rtProgramCallsiteSetPotentialCallees` (

```
RTprogram program,
const char * name,
const int * ids,
int numIds )
```

Sets the program ids that may potentially be called at a call site.

Description

`rtProgramCallsiteSetPotentialCallees` specifies the program IDs of potential callees at the call site in the *program* identified by *name* to the list provided in *ids*. If *program* is not a valid *RTprogram* or the *program* does not contain a call site with the identifier *name* or *ids* contains invalid program ids, returns `RT_ERROR_INVALID_VALUE`.

Parameters

in	<i>program</i>	The program that includes the call site.
in	<i>name</i>	The string identifier for the call site to modify.
in	<i>ids</i>	The program IDs of the programs that may potentially be called at the call site
in	<i>numIds</i>	The size of the array passed in for <i>ids</i> .

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtProgramCallsiteSetPotentialCallees` was introduced in OptiX 6.0.

See also [rtProgramGetId](#)

5.13.2.3 RTResult RTAPI rtProgramCreateFromPTXFile (

```
RTcontext context,
const char * filename,
const char * programName,
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 *programName*.

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>programName</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](#)

5.13.2.4 RTResult RTAPI rtProgramCreateFromPTXFiles (

```
RTcontext context,
unsigned int n,
const char ** filenames,
const char * programName,
RTprogram * program )
```

Creates a new program object.

Description

`rtProgramCreateFromPTXFiles` allocates and returns a handle to a new program object. The program is created by linking PTX code held in one or more files. C-style linking rules apply: global functions

and variables are visible across input files and must be defined uniquely. There must be a visible function for *programName*.

Parameters

in	<i>context</i>	The context to create the program in
in	<i>n</i>	Number of filenames
in	<i>filenames</i>	Array of one or more paths to files containing PTX code
in	<i>programName</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

See also [RT_PROGRAM](#), [rtProgramCreateFromPTXString](#), [rtProgramCreateFromPTXStrings](#), [rtProgramCreateFromPTXFile](#), [rtProgramDestroy](#)

5.13.2.5 RTResult RTAPI rtProgramCreateFromPTXString (

```
    RTcontext context,
    const char * ptx,
    const char * programName,
    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 *programName*.

Parameters

in	<i>context</i>	The context to create the program in
in	<i>ptx</i>	The string containing the PTX code
in	<i>programName</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`, `rtProgramCreateFromPTXFiles`, `rtProgramCreateFromPTXStrings`, `rtProgramDestroy`

5.13.2.6 RTResult RTAPI `rtProgramCreateFromPTXStrings` (

```
RTcontext context,
unsigned int n,
const char ** ptxStrings,
const char * programName,
RTprogram * program )
```

Creates a new program object.

Description

`rtProgramCreateFromPTXStrings` allocates and returns a handle to a new program object. The program is created by linking PTX code held in one or more *NULL-terminated* strings. C-style linking rules apply: global functions and variables are visible across input strings and must be defined uniquely. There must be a visible function for *programName*.

Parameters

in	<i>context</i>	The context to create the program in
in	<i>n</i>	Number of ptx strings
in	<i>ptxStrings</i>	Array of strings containing PTX code
in	<i>programName</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

See also [RT_PROGRAM](#), [rtProgramCreateFromPTXFile](#), [rtProgramCreateFromPTXFiles](#), [rtProgramCreateFromPTXString](#), [rtProgramDestroy](#)

5.13.2.7 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](#)

5.13.2.8 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](#)

5.13.2.9 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](#)

5.13.2.10 RTResult RTAPI rtProgramGetId (

RTprogram *program*,

```
int * programId )
```

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 *programId* 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>programId</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](#)

5.13.2.11 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`

5.13.2.12 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`

5.13.2.13 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`

5.13.2.14 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`

5.13.2.15 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`

5.14 Buffer functions

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)
- RTresult RTAPI rtBufferCreateFromGLBO (RTcontext context, unsigned int bufferdesc, unsigned int glld, RTbuffer *buffer)
- RTresult RTAPI rtBufferGetGLBOId (RTbuffer buffer, unsigned int *glld)
- RTresult RTAPI rtBufferGLRegister (RTbuffer buffer)
- RTresult RTAPI rtBufferGLUnregister (RTbuffer buffer)
- 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 elementSize)
- RTresult RTAPI rtBufferGetElementSize (RTbuffer buffer, RTsize *elementSize)
- 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 **userPointer)
- RTresult RTAPI rtBufferUnmap (RTbuffer buffer)
- RTresult RTAPI rtBufferMapEx (RTbuffer buffer, unsigned int mapFlags, unsigned int level, void *userOwned, void **optixOwned)
- RTresult RTAPI rtBufferUnmapEx (RTbuffer buffer, unsigned int level)

- RTresult RTAPI rtBufferGetId (RTbuffer buffer, int *bufferId)
- RTresult RTAPI rtBufferGetProgressiveUpdateReady (RTbuffer buffer, int *ready, unsigned int *subframeCount, unsigned int *maxSubframes)
- RTresult RTAPI rtBufferBindProgressiveStream (RTbuffer stream, RTbuffer source)
- RTresult RTAPI rtBufferSetAttribute (RTbuffer buffer, RTbufferattribute attrib, RTsize size, const void *p)
- RTresult RTAPI rtBufferGetAttribute (RTbuffer buffer, RTbufferattribute attrib, RTsize size, void *p)

5.14.1 Detailed Description

Functions related to an OptiX Buffer.

5.14.2 Function Documentation

5.14.2.1 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`

5.14.2.2 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`
- `RT_BUFFER_DISCARD_HOST_MEMORY`

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.

The flag `RT_BUFFER_DISCARD_HOST_MEMORY` can only be used in combination with `RT_BUFFER_INPUT`. The data will be synchronized to the devices as soon as the buffer is unmapped from the host using `rtBufferUnmap` or `rtBufferUnmapEx` and the memory allocated on the host will be deallocated. It is preferred to map buffers created with the `RT_BUFFER_DISCARD_HOST_MEMORY` using `rtBufferMapEx` with the `RT_BUFFER_MAP_WRITE_DISCARD` option enabled. If it is mapped

using `rtBufferMap` or the `RT_BUFFER_MAP_WRITE` option instead, the data needs to be synchronized to the host during mapping. Note that the data that is allocated on the devices will not be deallocated until the buffer is destroyed.

Returns `RT_ERROR_INVALID_VALUE` if `buffer` is `NULL`.

Parameters

in	<code>context</code>	The context to create the buffer in
in	<code>bufferdesc</code>	Bitwise <i>or</i> combination of the <code>type</code> and <code>flags</code> of the new buffer
out	<code>buffer</code>	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`

5.14.2.3 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	<code>context</code>	The context to create the buffer in
in	<code>bufferdesc</code>	Bitwise <i>or</i> combination of the <code>type</code> and <code>flags</code> of the new buffer
out	<code>buffer</code>	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`

5.14.2.4 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 its devices. Currently no *flags* are supported for interop buffers.

Parameters

in	<code>context</code>	The context to create the buffer in
in	<code>bufferdesc</code>	Bitwise or combination of the <i>type</i> and <i>flags</i> of the new buffer
in	<code>glld</code>	The OpenGL image object resource handle for use in OptiX
out	<code>buffer</code>	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](#)

5.14.2.5 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](#)

5.14.2.6 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`

5.14.2.7 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](#)

5.14.2.8 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](#)

5.14.2.9 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`

5.14.2.10 RTResult RTAPI `rtBufferGetElementSize` (

`RTbuffer buffer,`
`RTsize * elementSize)`

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 **elementSize*. Returns `RT_ERROR_INVALID_VALUE` if given a *NULL* pointer.

Parameters

in	<i>buffer</i>	Specifies the buffer to be queried
out	<i>elementSize</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`

5.14.2.11 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`

5.14.2.12 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`

5.14.2.13 RTResult RTAPI `rtBufferGetId` (

```
RTbuffer buffer,  
int * bufferId )
```

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 `*bufferId` 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>bufferId</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`

5.14.2.14 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
----	---------------	--

Parameters

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`, `rtBufferGetSize1D`, `rtBufferGetSize2D`, `rtBufferGetSize3D`, `rtBufferSetSizev`, `rtBufferGetMipLevelSize1D`, `rtBufferGetMipLevelSize2D`, `rtBufferGetMipLevelSize3D`, `rtBufferGetSize1D`, `rtBufferGetSize2D`, `rtBufferGetSize3D`, `rtBufferGetSizev`

5.14.2.15 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`, `rtBufferGetSize1D`, `rtBufferGetSize2D`, `rtBufferGetSize3D`, `rtBufferSetSizev`, `rtBufferGetMipLevelSize2D`, `rtBufferGetMipLevelSize3D`, `rtBufferGetMipLevelCount`, `rtBufferGetSize1D`, `rtBufferGetSize2D`, `rtBufferGetSize3D`, `rtBufferGetSizev`

5.14.2.16 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`

5.14.2.17 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
----	---------------	---

Parameters

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`, `rtBufferGetSize1D`, `rtBufferGetSize2D`, `rtBufferGetSize3D`, `rtBufferGetSizev`, `rtBufferGetMipLevelSize1D`, `rtBufferGetMipLevelSize2D`, `rtBufferGetMipLevelCount`, `rtBufferGetSize1D`, `rtBufferGetSize2D`, `rtBufferGetSize3D`, `rtBufferGetSizev`

5.14.2.18 RTResult RTAPI `rtBufferGetProgressiveUpdateReady` (

```
RTbuffer buffer,
int * ready,
unsigned int * subframeCount,
unsigned int * maxSubframes )
```

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 *subframeCount* and *maxSubframes* 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>subframeCount</i>	The number of subframes accumulated in the latest result
out	<i>maxSubframes</i>	The <i>maxSubframes</i> parameter as specified in the call to <code>rtContextLaunchProgressive2D</code>

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtBufferGetProgressiveUpdateReady` was introduced in OptiX 3.8.

See also `rtContextLaunchProgressive2D`

5.14.2.19 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`

5.14.2.20 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`

5.14.2.21 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`

5.14.2.22 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`

5.14.2.23 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

in	<i>buffer</i>	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`

5.14.2.24 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

in	<i>buffer</i>	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`

5.14.2.25 RTResult RTAPI `rtBufferMap` (

```
RTbuffer buffer,
void ** userPointer )
```

Maps a buffer object to the host.

Description

`rtBufferMap` returns a pointer, accessible by the host, in `*userPointer` that contains a mapped copy of the contents of `buffer`. The memory pointed to by `*userPointer` 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);
rtBufferSetSize1D(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	<code>buffer</code>	The buffer to be mapped
out	<code>userPointer</code>	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`

5.14.2.26 RTResult RTAPI `rtBufferMapEx` (

```
RTbuffer buffer,
unsigned int mapFlags,
unsigned int level,
void * userOwned,
void ** optixOwned )
```

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 `*optixOwned`, or by copying the contents to a memory location pointed to by `userOwned`. 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 `mapFlags = RT_BUFFER_MAP_READ_WRITE` and `level = 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>mapFlags</i>	Map flags, see below
in	<i>level</i>	The mipmap level to be mapped
in	<i>userOwned</i>	Not yet supported. Must be NULL
out	<i>optixOwned</i>	Return handle to a user pointer where the buffer will be mapped to

The following flags are supported for `mapFlags`. 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](#)

5.14.2.27 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](#) functions, 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](#)

5.14.2.28 RTResult RTAPI rtBufferSetAttribute (

RTbuffer *buffer*,

RTbufferattribute *attrib*,

RTsize *size*,

const 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	<code>buffer</code>	The buffer on which to set the attribute
in	<code>attrib</code>	The attribute to set
in	<code>size</code>	The size of the attribute value, in bytes
in	<code>p</code>	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`

5.14.2.29 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`

5.14.2.30 RTResult RTAPI `rtBufferSetElementSize` (

`RTbuffer buffer,`
`RTsize elementSize)`

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 *elementSize* and should not be 0. If the buffer has format `RT_FORMAT_USER`, and *elementSize* is not 0, then the buffer's individual element size is set to *elementSize* 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 *elementSize* is 0.

Parameters

in	<i>buffer</i>	Specifies the buffer to be modified
in	<i>elementSize</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`

5.14.2.31 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_LONG_LONG
- RT_FORMAT_LONG_LONG2
- RT_FORMAT_LONG_LONG3
- RT_FORMAT_LONG_LONG4
- RT_FORMAT_UNSIGNED_LONG_LONG
- RT_FORMAT_UNSIGNED_LONG_LONG2
- RT_FORMAT_UNSIGNED_LONG_LONG3
- RT_FORMAT_UNSIGNED_LONG_LONG4
- RT_FORMAT_UNSIGNED_BC1
- RT_FORMAT_UNSIGNED_BC2
- RT_FORMAT_UNSIGNED_BC3
- RT_FORMAT_UNSIGNED_BC4
- RT_FORMAT_BC4
- RT_FORMAT_UNSIGNED_BC5
- RT_FORMAT_BC5
- RT_FORMAT_UNSIGNED_BC6H
- RT_FORMAT_BC6H
- RT_FORMAT_UNSIGNED_BC7
- RT_FORMAT_USER

Buffers of block-compressed formats like RT_FORMAT_BC6H must be sized to a quarter of the uncompressed view resolution in each dimension, i.e.

```
rtBufferSetSize2D( buffer, width/4, height/4 );
```

The base type of the internal buffer will then correspond to RT_FORMAT_UNSIGNED_INT2 for BC1 and BC4 formats and RT_FORMAT_UNSIGNED_INT4 for all other BC formats.

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`

5.14.2.32 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	<code>buffer</code>	The buffer to be resized
in	<code>levels</code>	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`

5.14.2.33 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`

5.14.2.34 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`

5.14.2.35 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`

5.14.2.36 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`

5.14.2.37 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`

5.14.2.38 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
in	<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`

5.14.2.39 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

in	<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`

5.15 TextureSampler functions

Functions

- RTresult RTAPI rtTextureSamplerCreateFromGLImage (RTcontext context, unsigned int glId, RTgttarget target, RTtexturesampler *textureSampler)
- RTresult RTAPI rtTextureSamplerGetGLImageId (RTtexturesampler textureSampler, unsigned int *glId)
- RTresult RTAPI rtTextureSamplerGLRegister (RTtexturesampler textureSampler)
- RTresult RTAPI rtTextureSamplerGLUnregister (RTtexturesampler textureSampler)
- 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 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 *textureId)

5.15.1 Detailed Description

Functions related to an OptiX Texture Sampler.

5.15.2 Function Documentation

5.15.2.1 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	<code>context</code>	The context the texture sampler object will be created in
out	<code>texturesampler</code>	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](#)

5.15.2.2 RTResult RTAPI rtTextureSamplerCreateFromGLImage (

```
RTcontext context,
unsigned int gild,
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 [OptiX Interoperability Types](#) 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](#)

5.15.2.3 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](#)

5.15.2.4 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](#)

5.15.2.5 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](#)

5.15.2.6 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	<code>textureSampler</code>	The texture sampler object to be queried
out	<code>minification</code>	The return handle for the minification filtering mode of the texture sampler
out	<code>magnification</code>	The return handle for the magnification filtering mode of the texture sampler
out	<code>mipmapping</code>	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`

5.15.2.7 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	<code>textureSampler</code>	The texture sampler to be queried for its OpenGL buffer object id
in	<code>glId</code>	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`

5.15.2.8 RTResult RTAPI `rtTextureSamplerGetId` (

```
RTtexturesampler texturesampler,
int * textureId )
```

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 *textureId* is *NULL*, returns `RT_ERROR_INVALID_VALUE`.

Parameters

in	<i>texturesampler</i>	The texture sampler object to be queried for its ID
out	<i>textureId</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`

5.15.2.9 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](#)

5.15.2.10 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](#)

5.15.2.11 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](#)

5.15.2.12 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](#)

5.15.2.13 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](#)

5.15.2.14 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`

5.15.2.15 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

in	<code>textureSampler</code>	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`

5.15.2.16 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

in	<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](#)

```
5.15.2.17 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](#)

5.15.2.18 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 *texturesampler*. `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

<code>in</code>	<i>texturesampler</i>	The texture sampler object to be changed
<code>in</code>	<i>minification</i>	The new minification filter mode of the texture sampler
<code>in</code>	<i>magnification</i>	The new magnification filter mode of the texture sampler
<code>in</code>	<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`

5.15.2.19 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`

5.15.2.20 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`

5.15.2.21 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`

5.15.2.22 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

Parameters

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](#)

5.15.2.23 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`

5.15.2.24 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](#)

5.15.2.25 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](#)

5.16 Variable functions

Modules

- Variable setters
- Variable getters

Functions

- RTresult RTAPI rtVariableSetObject (RTvariable v, RTobject object)
- RTresult RTAPI rtVariableSetUserData (RTvariable v, RTsize size, const void *ptr)
- RTresult RTAPI rtVariableGetObject (RTvariable v, RTobject *object)
- RTresult RTAPI rtVariableGetUserData (RTvariable v, RTsize size, void *ptr)
- RTresult RTAPI rtVariableGetName (RTvariable v, const char **nameReturn)
- RTresult RTAPI rtVariableGetAnnotation (RTvariable v, const char **annotationReturn)
- RTresult RTAPI rtVariableGetType (RTvariable v, RTobjecttype *typeReturn)
- RTresult RTAPI rtVariableGetContext (RTvariable v, RTcontext *context)
- RTresult RTAPI rtVariableGetSize (RTvariable v, RTsize *size)

5.16.1 Detailed Description

Functions related to variable handling.

5.16.2 Function Documentation

5.16.2.1 RTResult RTAPI rtVariableGetAnnotation (

RTvariable v,
const char ** annotationReturn)

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 `*annotationReturn`. If `v` is not a valid variable, this call sets `*annotationReturn` to `NULL` and returns `RT_ERROR_INVALID_VALUE`. `*annotationReturn` will point to valid memory until another API function that returns a string is called.

Parameters

in	<code>v</code>	Specifies the program variable to be queried
out	<code>annotationReturn</code>	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`

5.16.2.2 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	<code>v</code>	Specifies the program variable to be queried
out	<code>context</code>	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`

5.16.2.3 RTresult RTAPI `rtVariableGetName` (

`RTvariable v,`
`const char ** nameReturn)`

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 `*nameReturn`. If `v` is not a valid variable, this call sets `*nameReturn` to `NULL` and returns `RT_ERROR_INVALID_VALUE`. `*nameReturn` 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>nameReturn</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`

5.16.2.4 RTresult RTAPI `rtVariableGetObject` (

`RTvariable v,`
`RTobject * 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 `RTobject` 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](#)

5.16.2.5 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	<i>v</i>	Specifies the program variable to be queried
out	<i>size</i>	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](#)

5.16.2.6 RTResult RTAPI rtVariableGetType (

```
RTvariable v,
RTobjecttype * typeReturn )
```

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 **typeReturn*, 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 `*typeReturn` to `RT_OBJECTTYPE_UNKNOWN` if `v` is not a valid variable. Returns `RT_ERROR_INVALID_VALUE` if given a `NULL` pointer.

Parameters

in	<code>v</code>	Specifies the program variable to be queried
out	<code>typeReturn</code>	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](#)

5.16.2.7 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`

5.16.2.8 RTResult RTAPI `rtVariableSetObject` (

```
RTvariable v,
RTobject 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`

5.16.2.9 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	<code>v</code>	Specifies the program variable to be modified
in	<code>size</code>	Specifies the size of the new value, in bytes
in	<code>ptr</code>	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`

5.17 Variable setters

- 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 rtVariableSet1ll (RTvariable v, long long ll1)
- RTresult RTAPI rtVariableSet2ll (RTvariable v, long long ll1, long long ll2)
- RTresult RTAPI rtVariableSet3ll (RTvariable v, long long ll1, long long ll2, long long ll3)
- RTresult RTAPI rtVariableSet4ll (RTvariable v, long long ll1, long long ll2, long long ll3, long long ll4)
- RTresult RTAPI rtVariableSet1llv (RTvariable v, const long long **ll)
- RTresult RTAPI rtVariableSet2llv (RTvariable v, const long long **ll)
- RTresult RTAPI rtVariableSet3llv (RTvariable v, const long long **ll)
- RTresult RTAPI rtVariableSet4llv (RTvariable v, const long long **ll)
- RTresult RTAPI rtVariableSet1ull (RTvariable v, unsigned long long ull1)
- RTresult RTAPI rtVariableSet2ull (RTvariable v, unsigned long long ull1, unsigned long long ull2)
- RTresult RTAPI rtVariableSet3ull (RTvariable v, unsigned long long ull1, unsigned long long ull2, unsigned long long ull3)
- RTresult RTAPI rtVariableSet4ull (RTvariable v, unsigned long long ull1, unsigned long long ull2, unsigned long long ull3, unsigned long long ull4)
- RTresult RTAPI rtVariableSet1ullv (RTvariable v, const unsigned long long *ull)
- RTresult RTAPI rtVariableSet2ullv (RTvariable v, const unsigned long long *ull)
- RTresult RTAPI rtVariableSet3ullv (RTvariable v, const unsigned long long *ull)
- RTresult RTAPI rtVariableSet4ullv (RTvariable v, const unsigned long long *ull)

- 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)

5.17.1 Detailed Description

Functions designed to modify the value of a program variable.

5.17.2 Function Documentation

5.17.2.1 RTResult RTAPI rtVariableSet1f (

RTvariable v,
float f1)

Functions designed to modify the value of a program variable.

Description

Variable setters 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 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

Variable setters were introduced in OptiX 1.0.

See also [Variable getters](#), [Variable setters](#), [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

5.17.2.2 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

5.17.2.3 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

5.17.2.4 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

5.17.2.5 RTResult RTAPI rtVariableSet1ll (

RTvariable *v*,
long long *ll1*)

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>ll1</i>	Specifies the new long long value of the program variable

5.17.2.6 RTResult RTAPI rtVariableSet1llv (

RTvariable *v*,
const long long * *ll*)

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>ll</i>	Array of long long values to set the variable to

5.17.2.7 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

5.17.2.8 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

5.17.2.9 RTResult RTAPI rtVariableSet1ull (

RTvariable *v*,
unsigned long long *ull1*)

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>ull1</i>	Specifies the new unsigned long long value of the program variable

5.17.2.10 RTResult RTAPI rtVariableSet1ullv (

```
RTvariable v,
const unsigned long long * ull )
```

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>ull</i>	Array of unsigned long long values to set the variable to

5.17.2.11 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

5.17.2.12 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

5.17.2.13 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

5.17.2.14 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

5.17.2.15 RTResult RTAPI rtVariableSet2ll (

RTvariable v,
long long II1,
long long II2)

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>II1</i>	Specifies the new long long value of the program variable
in	<i>II2</i>	Specifies the new long long value of the program variable

5.17.2.16 RTResult RTAPI rtVariableSet2llv (

RTvariable v,
const long long * II)

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>II</i>	Array of long long values to set the variable to

5.17.2.17 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

5.17.2.18 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

5.17.2.19 RTResult RTAPI rtVariableSet2ull (

RTvariable *v*,
unsigned long long *ull1*,
unsigned long long *ull2*)

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>ull1</i>	Specifies the new unsigned long long value of the program variable
in	<i>ull2</i>	Specifies the new unsigned long long value of the program variable

5.17.2.20 RTResult RTAPI rtVariableSet2ullv (

RTvariable *v*,
const unsigned long long * *ull*)

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>ull</i>	Array of unsigned long long values to set the variable to

5.17.2.21 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

5.17.2.22 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

5.17.2.23 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

5.17.2.24 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

5.17.2.25 RTResult RTAPI rtVariableSet3ll (

```
RTvariable v,
long long ll1,
long long ll2,
long long ll3 )
```

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>ll1</i>	Specifies the new long long value of the program variable
in	<i>ll2</i>	Specifies the new long long value of the program variable
in	<i>ll3</i>	Specifies the new long long value of the program variable

5.17.2.26 RTResult RTAPI rtVariableSet3llv (

```
RTvariable v,
const long long * ll )
```

Parameters

in	v	Specifies the program variable to be modified
in	ll	Array of long long values to set the variable to

5.17.2.27 RTResult RTAPI rtVariableSet3ui (

```
RTvariable v,
unsigned int u1,
unsigned int u2,
unsigned int u3 )
```

Parameters

in	v	Specifies the program variable to be modified
in	u1	Specifies the new unsigned integer value of the program variable
in	u2	Specifies the new unsigned integer value of the program variable
in	u3	Specifies the new unsigned integer value of the program variable

5.17.2.28 RTResult RTAPI rtVariableSet3uiv (

```
RTvariable v,
const unsigned int * u )
```

Parameters

in	v	Specifies the program variable to be modified
in	u	Array of unsigned integer values to set the variable to

5.17.2.29 RTResult RTAPI rtVariableSet3ull (

```
RTvariable v,
unsigned long long ull1,
unsigned long long ull2,
unsigned long long ull3 )
```

Parameters

in	v	Specifies the program variable to be modified
in	ull1	Specifies the new unsigned long long value of the program variable
in	ull2	Specifies the new unsigned long long value of the program variable
in	ull3	Specifies the new unsigned long long value of the program variable

5.17.2.30 RTresult RTAPI rtVariableSet3ullv (

RTvariable v,

const unsigned long long * ull)

Parameters

in	v	Specifies the program variable to be modified
in	ull	Array of unsigned long long values to set the variable to

5.17.2.31 RTresult RTAPI rtVariableSet4f (

RTvariable v,

float f1,

float f2,

float f3,

float f4)

Parameters

in	v	Specifies the program variable to be modified
in	f1	Specifies the new float value of the program variable
in	f2	Specifies the new float value of the program variable
in	f3	Specifies the new float value of the program variable
in	f4	Specifies the new float value of the program variable

5.17.2.32 RTresult RTAPI rtVariableSet4fv (

RTvariable v,

const float * f)

Parameters

in	v	Specifies the program variable to be modified
in	f	Array of float values to set the variable to

5.17.2.33 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

5.17.2.34 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

5.17.2.35 RTResult RTAPI rtVariableSet4ll (

RTvariable *v*,
long long *l1*,
long long *l2*,
long long *l3*,
long long *l4*)

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>l1</i>	Specifies the new long long value of the program variable
in	<i>l2</i>	Specifies the new long long value of the program variable
in	<i>l3</i>	Specifies the new long long value of the program variable
in	<i>l4</i>	Specifies the new long long value of the program variable

5.17.2.36 RTResult RTAPI rtVariableSet4llv (

RTvariable *v*,
const long long * *l*)

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>l</i>	Array of long long values to set the variable to

5.17.2.37 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

5.17.2.38 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

5.17.2.39 RTResult RTAPI rtVariableSet4ull (

```
RTvariable v,
unsigned long long ull1,
unsigned long long ull2,
unsigned long long ull3,
unsigned long long ull4 )
```

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>ull1</i>	Specifies the new unsigned long long value of the program variable
in	<i>ull2</i>	Specifies the new unsigned long long value of the program variable
in	<i>ull3</i>	Specifies the new unsigned long long value of the program variable
in	<i>ull4</i>	Specifies the new unsigned long long value of the program variable

5.17.2.40 RTResult RTAPI rtVariableSet4ullv (

```
RTvariable v,
```

const unsigned long long * *ull*)

Parameters

in	<i>v</i>	Specifies the program variable to be modified
in	<i>ull</i>	Array of unsigned long long values to set the variable to

5.17.2.41 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

5.17.2.42 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

5.17.2.43 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

5.17.2.44 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

5.17.2.45 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

5.17.2.46 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

5.17.2.47 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

5.17.2.48 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

5.17.2.49 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

5.18 Variable getters

- 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 rtVariableGet1ll (RTvariable v, long long *ll1)
- RTresult RTAPI rtVariableGet2ll (RTvariable v, long long *ll1, long long *ll2)
- RTresult RTAPI rtVariableGet3ll (RTvariable v, long long *ll1, long long *ll2, long long *ll3)
- RTresult RTAPI rtVariableGet4ll (RTvariable v, long long *ll1, long long *ll2, long long *ll3, long long *ll4)
- RTresult RTAPI rtVariableGet1llv (RTvariable v, long long *ll)
- RTresult RTAPI rtVariableGet2llv (RTvariable v, long long *ll)
- RTresult RTAPI rtVariableGet3llv (RTvariable v, long long *ll)
- RTresult RTAPI rtVariableGet4llv (RTvariable v, long long *ll)
- RTresult RTAPI rtVariableGet1ull (RTvariable v, unsigned long long *u1)
- RTresult RTAPI rtVariableGet2ull (RTvariable v, unsigned long long *u1, unsigned long long *u2)
- RTresult RTAPI rtVariableGet3ull (RTvariable v, unsigned long long *u1, unsigned long long *u2, unsigned long long *u3)
- RTresult RTAPI rtVariableGet4ull (RTvariable v, unsigned long long *u1, unsigned long long *u2, unsigned long long *u3, unsigned long long *u4)
- RTresult RTAPI rtVariableGet1ullv (RTvariable v, unsigned long long *ull)
- RTresult RTAPI rtVariableGet2ullv (RTvariable v, unsigned long long *ull)
- RTresult RTAPI rtVariableGet3ullv (RTvariable v, unsigned long long *ull)

- RTresult RTAPI rtVariableGet4ullv (RTvariable v, unsigned long long *ull)
- 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)

5.18.1 Detailed Description

Functions designed to modify the value of a program variable.

5.18.2 Function Documentation

5.18.2.1 RTresult RTAPI rtVariableGet1f (

RTvariable v,
float * f1)

Functions designed to modify the value of a program variable.

Description

Variable getters 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

Variable getters were introduced in OptiX 1.0.

See also [Variable setters](#), [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

5.18.2.2 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

5.18.2.3 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

5.18.2.4 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

5.18.2.5 RTResult RTAPI rtVariableGet1ll (

```
RTvariable v,
```

long long * *l1*)

Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>l1</i>	Integer value to be returned

5.18.2.6 RTResult RTAPI rtVariableGet1llv (

RTvariable *v*,

long long * *l1*)

Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>l1</i>	Array of integer values to be returned

5.18.2.7 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

5.18.2.8 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

5.18.2.9 RTResult RTAPI rtVariableGet1ull (

RTvariable *v*,

unsigned long long * *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

5.18.2.10 RTResult RTAPI rtVariableGet1ullv (

RTvariable v,

unsigned long long * ull)

Parameters

in	v	Specifies the program variable whose value is to be returned
in	ull	Array of unsigned integer values to be returned

5.18.2.11 RTResult RTAPI rtVariableGet2f (

RTvariable v,

float * f1,

float * f2)

Parameters

in	v	Specifies the program variable whose value is to be returned
in	f1	Float value to be returned
in	f2	Float value to be returned

5.18.2.12 RTResult RTAPI rtVariableGet2fv (

RTvariable v,

float * f)

Parameters

in	v	Specifies the program variable whose value is to be returned
in	f	Array of float value(s) to be returned

5.18.2.13 RTResult RTAPI rtVariableGet2i (

RTvariable v,

int * i1,

int * i2)

Parameters

in	v	Specifies the program variable whose value is to be returned
in	i1	Integer value to be returned
in	i2	Integer value to be returned

5.18.2.14 RTResult RTAPI rtVariableGet2iv (

RTvariable v,

int * i)

Parameters

in	v	Specifies the program variable whose value is to be returned
in	i	Array of integer values to be returned

5.18.2.15 RTResult RTAPI rtVariableGet2II (

RTvariable v,
long long * II1,
long long * II2)

Parameters

in	v	Specifies the program variable whose value is to be returned
in	II1	Integer value to be returned
in	II2	Integer value to be returned

5.18.2.16 RTResult RTAPI rtVariableGet2IIV (

RTvariable v,
long long * II)

Parameters

in	v	Specifies the program variable whose value is to be returned
in	II	Array of integer values to be returned

5.18.2.17 RTResult RTAPI rtVariableGet2ui (

RTvariable v,
unsigned int * u1,
unsigned int * u2)

Parameters

in	v	Specifies the program variable whose value is to be returned
in	u1	Unsigned integer value to be returned
in	u2	Unsigned integer value to be returned

5.18.2.18 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

5.18.2.19 RTResult RTAPI rtVariableGet2ull (

RTvariable *v*,
unsigned long long * *u1*,
unsigned long long * *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

5.18.2.20 RTResult RTAPI rtVariableGet2ullv (

RTvariable *v*,
unsigned long long * *ull*)

Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>ull</i>	Array of unsigned integer values to be returned

5.18.2.21 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

5.18.2.22 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

5.18.2.23 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

5.18.2.24 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

5.18.2.25 RTResult RTAPI rtVariableGet3ll (

```
RTvariable v,  
long long * ll1,  
long long * ll2,  
long long * ll3 )
```

Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>ll1</i>	Integer value to be returned
in	<i>ll2</i>	Integer value to be returned
in	<i>ll3</i>	Integer value to be returned

5.18.2.26 RTResult RTAPI rtVariableGet3llv (

```
RTvariable v,
long long * u1 )
```

Parameters

in	v	Specifies the program variable whose value is to be returned
in	u1	Array of integer values to be returned

5.18.2.27 RTResult RTAPI rtVariableGet3ui (

```
RTvariable v,
unsigned int * u1,
unsigned int * u2,
unsigned int * u3 )
```

Parameters

in	v	Specifies the program variable whose value is to be returned
in	u1	Unsigned integer value to be returned
in	u2	Unsigned integer value to be returned
in	u3	Unsigned integer value to be returned

5.18.2.28 RTResult RTAPI rtVariableGet3uiv (

```
RTvariable v,
unsigned int * u )
```

Parameters

in	v	Specifies the program variable whose value is to be returned
in	u	Array of unsigned integer values to be returned

5.18.2.29 RTResult RTAPI rtVariableGet3ull (

```
RTvariable v,
unsigned long long * u1,
unsigned long long * u2,
unsigned long long * u3 )
```

Parameters

in	v	Specifies the program variable whose value is to be returned
in	u1	Unsigned integer value to be returned
in	u2	Unsigned integer value to be returned
in	u3	Unsigned integer value to be returned

5.18.2.30 RTResult RTAPI rtVariableGet3ullv (

RTvariable *v*,

unsigned long long * *ull*)

Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>ull</i>	Array of unsigned integer values to be returned

5.18.2.31 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

5.18.2.32 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

5.18.2.33 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

5.18.2.34 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

5.18.2.35 RTResult RTAPI rtVariableGet4ll (

RTvariable *v*,
long long * *ll1*,
long long * *ll2*,
long long * *ll3*,
long long * *ll4*)

Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>ll1</i>	Integer value to be returned
in	<i>ll2</i>	Integer value to be returned
in	<i>ll3</i>	Integer value to be returned
in	<i>ll4</i>	Integer value to be returned

5.18.2.36 RTResult RTAPI rtVariableGet4llv (

RTvariable *v*,
long long * *ll*)

Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>ll</i>	Array of integer values to be returned

5.18.2.37 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

5.18.2.38 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

5.18.2.39 RTResult RTAPI rtVariableGet4ull (

```
RTvariable v,
unsigned long long * u1,
unsigned long long * u2,
unsigned long long * u3,
unsigned long long * 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

5.18.2.40 RTResult RTAPI rtVariableGet4ullv (

```
RTvariable v,
```

```
unsigned long long * ull )
```

Parameters

in	<i>v</i>	Specifies the program variable whose value is to be returned
in	<i>ull</i>	Array of unsigned integer values to be returned

5.18.2.41 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

5.18.2.42 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

5.18.2.43 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

5.18.2.44 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

5.18.2.45 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

5.18.2.46 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

5.18.2.47 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

5.18.2.48 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

5.18.2.49 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

5.19 CommandList functions

Functions

- RTresult RTAPI rtPostProcessingStageCreateBuiltin (RTcontext context, const char *builtinName, RTpostprocessingstage *stage)
- RTresult RTAPI rtPostProcessingStageDestroy (RTpostprocessingstage stage)
- RTresult RTAPI rtPostProcessingStageDeclareVariable (RTpostprocessingstage stage, const char *name, RTvariable *v)
- RTresult RTAPI rtPostProcessingStageGetContext (RTpostprocessingstage stage, RTcontext *context)
- RTresult RTAPI rtPostProcessingStageQueryVariable (RTpostprocessingstage stage, const char *name, RTvariable *variable)
- RTresult RTAPI rtPostProcessingStageGetVariableCount (RTpostprocessingstage stage, unsigned int *count)
- RTresult RTAPI rtPostProcessingStageGetVariable (RTpostprocessingstage stage, unsigned int index, RTvariable *variable)
- RTresult RTAPI rtCommandListCreate (RTcontext context, RTcommandlist *list)
- RTresult RTAPI rtCommandListDestroy (RTcommandlist list)
- RTresult RTAPI rtCommandListAppendPostprocessingStage (RTcommandlist list, RTpostprocessingstage stage, RTsize launchWidth, RTsize launchHeight)
- RTresult RTAPI rtCommandListAppendLaunch2D (RTcommandlist list, unsigned int entryPointIndex, RTsize launchWidth, RTsize launchHeight)
- RTresult RTAPI rtCommandListFinalize (RTcommandlist list)
- RTresult RTAPI rtCommandListExecute (RTcommandlist list)
- RTresult RTAPI rtCommandListGetContext (RTcommandlist list, RTcontext *context)

5.19.1 Detailed Description

Functions related to an OptiX Command List.

5.19.2 Function Documentation

5.19.2.1 RTresult RTAPI rtCommandListAppendLaunch2D (

RTcommandlist *list*,
unsigned int *entryPointIndex*,
RTsize *launchWidth*,
RTsize *launchHeight*)

Append a launch to the command list *list*.

Description

`rtCommandListAppendLaunch2D` appends a context launch to the command list *list*. It is invalid to call `rtCommandListAppendLaunch2D` after calling `rtCommandListFinalize`.

Parameters

in	<i>list</i>	Handle of the command list to append to
in	<i>entryPointIndex</i>	The initial entry point into the kernel
in	<i>launchWidth</i>	Width of the computation grid
in	<i>launchHeight</i>	Height of the computation grid

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtCommandListAppendLaunch2D` was introduced in OptiX 5.0.

See also `rtCommandListCreate`, `rtCommandListDestroy`, `rtCommandListAppendPostprocessingStage`, `rtCommandListFinalize`, `rtCommandListExecute`

5.19.2.2 RTResult RTAPI `rtCommandListAppendPostprocessingStage` (

`RTcommandlist list,`
`RTpostprocessingstage stage,`
`RTsize launchWidth,`
`RTsize launchHeight)`

Append a post-processing stage to the command list *list*.

Description

`rtCommandListAppendPostprocessingStage` appends a post-processing stage to the command list *list*. The command list must have been created from the same context as the the post-processing stage. The *launchWidth* and *launchHeight* specify the launch dimensions and may be different than the input or output buffers associated with each post-processing stage depending on the requirements of the post-processing stage appended. It is invalid to call `rtCommandListAppendPostprocessingStage` after calling `rtCommandListFinalize`.

NOTE: A post-processing stage can be added to multiple command lists or added to the same command list multiple times. Also note that destroying a post-processing stage will invalidate all command lists it was added to.

Parameters

in	<i>list</i>	Handle of the command list to append to
in	<i>stage</i>	The post-processing stage to append to the command list
in	<i>launchWidth</i>	This is a hint for the width of the launch dimensions to use for this stage. The stage can ignore this and use a suitable launch width instead.
in	<i>launchHeight</i>	This is a hint for the height of the launch dimensions to use for this stage. The stage can ignore this and use a suitable launch height instead.

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtCommandListAppendPostprocessingStage` was introduced in OptiX 5.0.

See also `rtCommandListCreate`, `rtCommandListDestroy`, `rtCommandListAppendLaunch2D`, `rtCommandListFinalize`, `rtCommandListExecute` `rtPostProcessingStageCreateBuiltin`,

5.19.2.3 RTResult RTAPI `rtCommandListCreate` (

`RTcontext context,`
`RTcommandlist * list)`

Creates a new command list.

Description

`rtCommandListCreate` creates a new command list. The *context* specifies the target context, and should be a value returned by `rtContextCreate`. The call sets **list* to the handle of a newly created list within *context*. Returns `RT_ERROR_INVALID_VALUE` if *list* is *NULL*.

A command list can be used to assemble a list of different types of commands and execute them later. At this point, commands can be built-in post-processing stages or context launches. Those are appended to the list using `rtCommandListAppendPostprocessingStage`, and `rtCommandListAppendLaunch2D`, respectively. Commands will be executed in the order they have been appended to the list. Thus later commands can use the results of earlier commands. Note that all commands added to the created list must be associated with the same *context*. It is invalid to mix commands from different contexts.

Parameters

in	<i>context</i>	Specifies the rendering context of the command list
out	<i>list</i>	New command list handle

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE
- RT_ERROR_MEMORY_ALLOCATION_FAILED

History

`rtCommandListCreate` was introduced in OptiX 5.0.

See also `rtCommandListDestroy`, `rtCommandListAppendPostprocessingStage`, `rtCommandListAppendLaunch2D`, `rtCommandListFinalize`, `rtCommandListExecute`

5.19.2.4 RTResult RTAPI rtCommandListDestroy (

RTcommandlist *list*)

Destroy a command list.

Description

`rtCommandListDestroy` destroys a command list from its context and deletes it. After the call, *list* is no longer a valid handle. Any stages associated with the command list are not destroyed.

Parameters

in	<i>list</i>	Handle of the command list to destroy
----	-------------	---------------------------------------

Return values

Relevant return values:

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

History

`rtCommandListDestroy` was introduced in OptiX 5.0.

See also [rtCommandListCreate](#), [rtCommandListAppendPostprocessingStage](#), [rtCommandListAppendLaunch2D](#), [rtCommandListFinalize](#), [rtCommandListExecute](#)

5.19.2.5 RTResult RTAPI rtCommandListExecute (

RTcommandlist *list*)

Execute the command list.

Description

`rtCommandListExecute` executes the command list. All added commands will be executed in the order in which they were added. Commands can access the results of earlier executed commands. This must be called after calling `rtCommandListFinalize`, otherwise an error will be returned and the command list is not executed. `rtCommandListExecute` can be called multiple times, but only one call may be active at the same time. Overlapping calls from multiple threads will result in undefined behavior.

Parameters

in	<i>list</i>	Handle of the command list to execute
----	-------------	---------------------------------------

Return values

Relevant return values:

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

History

`rtCommandListExecute` was introduced in OptiX 5.0.

See also [rtCommandListCreate](#), [rtCommandListDestroy](#), [rtCommandListAppendPostprocessingStage](#), [rtCommandListAppendLaunch2D](#), [rtCommandListFinalize](#),

5.19.2.6 RTResult RTAPI rtCommandListFinalize (

RTcommandlist *list*)

Finalize the command list.

This must be done before executing the command list.

Description

`rtCommandListFinalize` finalizes the command list. This will do all work necessary to prepare the command list for execution. Specifically it will do all work which can be shared between subsequent calls to `rtCommandListExecute`. It is invalid to call `rtCommandListExecute` before calling `rtCommandListFinalize`. It is invalid to call `rtCommandListAppendPostprocessingStage` or `rtCommandListAppendLaunch2D` after calling `finalize` and will result in an error. Also `rtCommandListFinalize` can only be called once on each command list.

Parameters

in	<i>list</i>	Handle of the command list to finalize
----	-------------	--

Return values

Relevant return values:

- `RT_SUCCESS`
- `RT_ERROR_INVALID_VALUE`

History

`rtCommandListFinalize` was introduced in OptiX 5.0.

See also [rtCommandListCreate](#), [rtCommandListDestroy](#), [rtCommandListAppendPostprocessingStage](#), [rtCommandListAppendLaunch2D](#), [rtCommandListExecute](#)

5.19.2.7 RTResult RTAPI rtCommandListGetContext (

RTcommandlist *list*,
RTcontext * *context*)

Returns the context associated with a command list.

Description

`rtCommandListGetContext` queries the context associated with a command list. The target command list is specified by *list*. The context of the command list is returned to **context* if the pointer *context* is not *NULL*. If *list* is not a valid command list, **context* is set to *NULL* and `RT_ERROR_INVALID_VALUE` is returned.

Parameters

in	<i>list</i>	Specifies the command list to be queried
out	<i>context</i>	Returns the context associated with the command list

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE

History

`rtCommandListGetContext` was introduced in OptiX 5.0.

See also [rtContextDeclareVariable](#)

5.19.2.8 RTResult RTAPI rtPostProcessingStageCreateBuiltin (

```
RTcontext context,
const char * builtinName,
RTpostprocessingstage * stage )
```

Creates a new post-processing stage.

Description

`rtPostProcessingStageCreateBuiltin` creates a new post-processing stage selected from a list of pre-defined post-processing stages. The *context* specifies the target context, and should be a value returned by `rtContextCreate`. Sets **stage* to the handle of a newly created stage within *context*.

Parameters

in	<i>context</i>	Specifies the rendering context to which the post-processing stage belongs
in	<i>builtinName</i>	The name of the built-in stage to instantiate
out	<i>stage</i>	New post-processing stage handle

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_CONTEXT
- RT_ERROR_INVALID_VALUE
- RT_ERROR_MEMORY_ALLOCATION_FAILED

History

`rtPostProcessingStageCreateBuiltin` was introduced in OptiX 5.0.

See also [rtPostProcessingStageDestroy](#), [rtPostProcessingStageGetContext](#), [rtPostProcessingStageQueryVariable](#), [rtPostProcessingStageGetVariableCount](#), [rtPostProcessingStageGetVariable](#)

5.19.2.9 RTResult RTAPI rtPostProcessingStageDeclareVariable (

```
RTpostprocessingstage stage,
const char * name,
```

RTvariable * v)

Declares a new named variable associated with a PostprocessingStage.

Description

`rtPostProcessingStageDeclareVariable` declares a new variable associated with a postprocessing stage. *stage* specifies the post-processing stage, and should be a value returned by `rtPostProcessingStageCreateBuiltin`. *name* specifies the name of the variable, and should be a *NULL-terminated* string. If there is currently no variable associated with *stage* named *name*, a new variable named *name* will be created and associated with *stage*. 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 `rtPostProcessingStageQueryVariable` or `rtPostProcessingStageGetVariable`. A declared variable does not have a type until its value is set with one of the `Variable` setters functions. Once a variable is set, its type cannot be changed anymore.

Parameters

in	<i>stage</i>	Specifies the associated postprocessing stage
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

`rtPostProcessingStageDeclareVariable` was introduced in OptiX 5.0.

See also `Variable` functions, `rtPostProcessingStageQueryVariable`, `rtPostProcessingStageGetVariable`

5.19.2.10 RTResult RTAPI `rtPostProcessingStageDestroy (RTpostprocessingstage stage)`

Destroy a post-processing stage.

Description

`rtPostProcessingStageDestroy` destroys a post-processing stage from its context and deletes it. The variables built into the stage are destroyed. After the call, *stage* is no longer a valid handle. After a post-processing stage was destroyed all command lists containing that stage are invalidated and can no longer be used.

Parameters

in	<i>stage</i>	Handle of the post-processing stage to destroy
----	--------------	--

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtPostProcessingStageDestroy` was introduced in OptiX 5.0.

See also `rtPostProcessingStageCreateBuiltin`, `rtPostProcessingStageGetContext`,
`rtPostProcessingStageQueryVariable`, `rtPostProcessingStageGetVariableCount`
`rtPostProcessingStageGetVariable`

5.19.2.11 RTResult RTAPI `rtPostProcessingStageGetContext` (

`RTpostprocessingstage stage,`
`RTcontext * context)`

Returns the context associated with a post-processing stage.

Description

`rtPostProcessingStageGetContext` queries a stage for its associated context. *stage* specifies the post-processing stage to query, and should be a value returned by `rtPostProcessingStageCreateBuiltin`. If both parameters are valid, **context* is set to the context associated with *stage*. Otherwise, the call has no effect and returns `RT_ERROR_INVALID_VALUE`.

Parameters

in	<i>stage</i>	Specifies the post-processing stage 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

`rtPostProcessingStageGetContext` was introduced in OptiX 5.0.

See also `rtPostProcessingStageCreateBuiltin`, `rtPostProcessingStageDestroy`,
`rtPostProcessingStageQueryVariable`, `rtPostProcessingStageGetVariableCount`
`rtPostProcessingStageGetVariable`

5.19.2.12 RTResult RTAPI `rtPostProcessingStageGetVariable` (

`RTpostprocessingstage stage,`
`unsigned int index,`

RTvariable * *variable*)

Returns a handle to a variable of a post-processing stage.

The variable is defined by index.

Description

`rtPostProcessingStageGetVariable` queries the handle of a post-processing stage's variable which is identified by its index . *stage* specifies the source post-processing stage, as returned by `rtPostProcessingStageCreateBuiltIn`. *index* specifies the index of the variable, and should be a less than the value return by `rtPostProcessingStageGetVariableCount`. If *index* is in the valid range, the call returns a handle to that variable in **variable*, otherwise *NULL*.

Parameters

in	<i>stage</i>	The post-processing stage to query the variable from
in	<i>index</i>	The index identifying the variable to be returned
out	<i>variable</i>	Returns the variable

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtPostProcessingStageGetVariable` was introduced in OptiX 5.0.

See also `rtPostProcessingStageCreateBuiltIn`, `rtPostProcessingStageDestroy`, `rtPostProcessingStageGetContext`, `rtPostProcessingStageQueryVariable`, `rtPostProcessingStageGetVariableCount`

5.19.2.13 RTResult RTAPI `rtPostProcessingStageGetVariableCount (`

`RTpostprocessingstage stage,`
`unsigned int * count)`

Returns the number of variables pre-defined in a post-processing stage.

Description

`rtPostProcessingStageGetVariableCount` returns the number of variables which are pre-defined in a post-processing stage. This can be used to iterate over the variables. Sets **count* to the number.

Parameters

in	<i>stage</i>	The post-processing stage to query the number of variables from
out	<i>count</i>	Returns the number of pre-defined variables

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtPostProcessingStageGetVariableCount` was introduced in OptiX 5.0.

See also `rtPostProcessingStageCreateBuiltin`, `rtPostProcessingStageDestroy`,
`rtPostProcessingStageGetContext`, `rtPostProcessingStageQueryVariable`,
`rtPostProcessingStageGetVariable`

5.19.2.14 RTResult RTAPI rtPostProcessingStageQueryVariable (
RTpostprocessingstage *stage*,
const char * *name*,
RTvariable * *variable*)

Returns a handle to a named variable of a post-processing stage.

Description

`rtPostProcessingStageQueryVariable` queries the handle of a post-processing stage's named variable. *stage* specifies the source post-processing stage, as returned by `rtPostProcessingStageCreateBuiltin`. *name* specifies the name of the variable, and should be a *NULL*-terminated string. If *name* is the name of a variable attached to *stage*, the call returns a handle to that variable in **variable*, otherwise *NULL*. Only pre-defined variables of that built-in stage type can be queried. It is not possible to add or remove variables.

Parameters

in	<i>stage</i>	The post-processing stage to query the variable from
in	<i>name</i>	The name that identifies the variable to be queried
out	<i>variable</i>	Returns the named variable

Return values

Relevant return values:

- RT_SUCCESS
- RT_ERROR_INVALID_VALUE

History

`rtPostProcessingStageQueryVariable` was introduced in OptiX 5.0.

See also `rtPostProcessingStageCreateBuiltin`, `rtPostProcessingStageDestroy`,
`rtPostProcessingStageGetContext`, `rtPostProcessingStageGetVariableCount`
`rtPostProcessingStageGetVariable`

5.20 Context-free functions

Functions

- RTresult RTAPI rtDeviceGetWGLDevice (int *device, HGPUNV gpu)
- RTresult RTAPI rtGetVersion (unsigned int *version)
- RTresult RTAPI rtGlobalSetAttribute (RTglobalattribute attrib, RTsize size, const void *p)
- RTresult RTAPI rtGlobalGetAttribute (RTglobalattribute attrib, RTsize size, void *p)
- RTresult RTAPI rtDeviceGetDeviceCount (unsigned int *count)
- RTresult RTAPI rtDeviceGetAttribute (int ordinal, RTdeviceattribute attrib, RTsize size, void *p)

5.20.1 Detailed Description

Functions that don't pertain to an OptiX context to be called.

5.20.2 Function Documentation

```
5.20.2.1 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)`
- `RT_DEVICE_ATTRIBUTE_PCI_BUS_ID` up to size-1, at most 13 chars
- `RT_DEVICE_ATTRIBUTE_COMPATIBLE_DEVICES` `sizeof(int)*(number of devices + 1)`

For `RT_DEVICE_ATTRIBUTE_COMPATIBLE_DEVICES`, the first *int* returned is the number of compatible device ordinals returned. A device is always compatible with itself, so the count will always be at least one. Size the output buffer based on the number of devices as returned by `rtDeviceGetDeviceCount`.

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. `RT_DEVICE_ATTRIBUTE_COMPATIBLE_DEVICES` was introduced in OptiX 6.0.

See also `rtDeviceGetDeviceCount`, `rtContextGetAttribute`

5.20.2.2 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	<i>count</i>	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`

5.20.2.3 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	<i>device</i>	A handle to the memory location where the OptiX device ordinal associated with <i>gpu</i> will be stored
in	<i>gpu</i>	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](#)

5.20.2.4 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](#)

5.20.2.5 RTresult RTAPI `rtGlobalGetAttribute` (

```
RTglobalattribute attrib,
RTsize size,
void * p )
```

Returns a global attribute.

Description

`rtGlobalGetAttribute` returns in *p* the value of the global attribute specified by *attrib*.

Each attribute can have a different size. The sizes are given in the following list:

- `RT_GLOBAL_ATTRIBUTE_ENABLE_RTX` `sizeof(int)`
- `RT_GLOBAL_ATTRIBUTE_DISPLAY_DRIVER_VERSION_MAJOR` `sizeof(unsigned int)`
- `RT_GLOBAL_ATTRIBUTE_DISPLAY_DRIVER_VERSION_MINOR` `sizeof(unsigned int)`

`RT_GLOBAL_ATTRIBUTE_ENABLE_RTX` is an experimental setting which sets the execution strategy used by Optix for the next context to be created.

`RT_GLOBAL_ATTRIBUTE_DISPLAY_DRIVER_VERSION_MAJOR` is an attribute to query the major version of the display driver found on the system. It's the first number in the driver version displayed as xxx.yy.

`RT_GLOBAL_ATTRIBUTE_DISPLAY_DRIVER_VERSION_MINOR` is an attribute to query the minor version of the display driver found on the system. It's the second number in the driver version displayed as xxx.yy.

Parameters

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_GLOBAL_ATTRIBUTE` - Can be returned if an unknown attribute was addressed.
- `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
- `RT_ERROR_DRIVER_VERSION_FAILED` - Can be returned if the display driver version could not be obtained.

History

`rtGlobalGetAttribute` was introduced in OptiX 5.1.

See also [rtGlobalSetAttribute](#),

5.20.2.6 RTresult RTAPI `rtGlobalSetAttribute` (

`RTglobalattribute attrib,`

`RTsize size,`

`const void * p)`

Set a global attribute.

Description

`rtGlobalSetAttribute` sets *p* as the value of the global attribute specified by *attrib*.

Each attribute can have a different size. The sizes are given in the following list:

- `RT_GLOBAL_ATTRIBUTE_ENABLE_RTX` `sizeof(int)`

`RT_GLOBAL_ATTRIBUTE_ENABLE_RTX` is an experimental attribute which sets the execution strategy used by Optix for the next context to be created. This attribute may be deprecated in a future release. Possible values: 0 (legacy default), 1 (compile and link programs separately).

Parameters

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_GLOBAL_ATTRIBUTE` - Can be returned if an unknown attribute was addressed.
- `RT_ERROR_INVALID_VALUE` - Can be returned if *size* does not match the proper size of the attribute, or if *p* is *NULL*

History

`rtGlobalSetAttribute` was introduced in OptiX 5.1.

See also [rtGlobalGetAttribute](#)

5.21 CUDA C Reference

Modules

- OptiX CUDA C declarations
- OptiX basic types
- OptiX CUDA C functions

5.21.1 Detailed Description

OptiX Functions related to host and device code.

5.22 OptiX CUDA C declarations

Macros

- `#define rtDeclareVariable(type, name, semantic, annotation)`
- `#define rtDeclareAnnotation(variable, annotation)`
- `#define rtCallableProgram(return_type, function_name,
parameter_list) rtDeclareVariable(optix::boundCallableProgramId<return_type parameter_list>,
function_name,);`
- `#define RT_PROGRAM __global__`
- `#define rtCallableProgramId optix::callableProgramId`
- `#define rtMarkedCallableProgramId optix::markedCallableProgramId`
- `#define rtCallableProgramX optix::boundCallableProgramId`

Functions

- `static __device__ float2 rtGetTriangleBarycentrics ()`

5.22.1 Detailed Description

Functions designed to declare programs and types used by OptiX device code.

5.22.2 Macro Definition Documentation

5.22.2.1 `#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`

5.22.2.2 `#define rtCallableProgram(return_type, function_name,`

```
parameter_list ) rtDeclareVariable(optix::boundCallableProgramId<return_type
parameter_list>, function_name,,);
```

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

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`

5.22.2.3 #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` `rtMarkedCallableProgramId`

5.22.2.4 #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`

5.22.2.5 #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](#)

5.22.2.6 #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 *semanticName*, 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. The vector is *not* guaranteed to be normalized. 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

Parameters

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`

5.22.2.7 #define rtMarkedCallableProgramId optix::markedCallableProgramId

Marked Callable Program ID Declaration.

Description

`rtMarkedCallableProgramId` declares callable program *name*, which will appear to be a callable function with the specified return type and list of arguments. Calls to this callable program can be referenced on the host by the given *callSiteName* in order to specify the set of callable programs that may be called at a specific call site. This allows to use bindless callable programs that call `rtTrace`. Callable programs that call `rtTrace` need a different call semantic than programs that do not. Specifying the callable programs that may potentially be called at a call site allow OptiX to determine the correct call semantics at each call site. Programs that are declared using `rtCallableProgramId` may only call trace if they are used in an `rtVariable` or in a `rtBuffer` of type `rtCallableProgramId`. The `rtMarkedCallableProgram` type is only available on the device and cannot be used in an `rtVariable`. Objects of type `rtCallableProgramId` can be transformed into `rtMarkedCallableProgramId` by using the appropriate constructor.

Example(s):

```
// Uses named call site marking, potential callees can be set through the host API,
// needed call semantics will determined based on those.
rtMarkedCallableProgramId<float3(float3, float)> modColor(id, "modColorCall");

// callable1 cannot call rtTrace
rtCallableProgramId<void(void)> callable1(id);
// Create marked callable from callable1. Uses named call site marking.
rtMarkedCallableProgramId<void(void)> markedCallable1(callable1, "callSite1");

// Variables of type rtCallableProgramId use automatic detection of the needed call semantics.
rtDeclareVariable(rtCallableProgramId<void(void)>, callable, , );
callable();

// Buffers of type rtCallableProgramId use automatic detection of the needed call semantics.
rtBuffer<rtCallableProgramId<void(void)>, 1> programBuffer;
```

```
programBuffer[0]();  
// Overwrite automatic marking with named marking  
rtMarkedCallableProgramId<void(void)> marked(programBuffer[0], "callSite2");  
// Use information provided through host API to determine call semantics.  
marked();
```

History

`rtCallableProgramId` was introduced in OptiX 6.0.

See also [rtCallableProgram](#)

5.22.3 Function Documentation

5.22.3.1 static __device__ float2 rtGetTriangleBarycentrics() [inline], [static]

Accessor for barycentrics for built in triangle intersection.

Description

`rtGetTriangleBarycentrics` returns the barycentric coordinates of the intersected triangle. This function is only accessible in a program attached as an attribute program to an `RTgeometrytriangles` object.

History

- `rtGetTriangleBarycentrics` was introduced in OptiX 6.0.

See also [rtGeometryTrianglesSetAttributeProgram](#)

5.23 OptiX basic types

Classes

- struct Ray
- struct rtObject
- class optix::Aabb
- class optix::Matrix< M, N >
- class optix::Quaternion

Macros

- #define rtBuffer __device__ optix::buffer
- #define rtBufferId optix::bufferId
- #define rtTextureSampler texture

5.23.1 Detailed Description

Basic types used in OptiX.

5.23.2 Macro Definition Documentation

5.23.2.1 #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`

5.23.2.2 #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`

5.23.2.3 #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`

5.24 OptiX CUDA C functions

Modules

- Texture fetch functions
- `rtPrintf` functions

Functions

- template<class T>


```
static __device__ void rtTrace(rtObject topNode, optix::Ray ray, T &prd, RTvisibilitymask mask=RT_VISIBILITY_ALL, RTrayflags flags=RT_RAY_FLAG_NONE)
```
- 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__ unsigned int `rtGetPrimitiveIndex` ()
- static __device__ bool `rtIsTriangleHit` ()
- static __device__ bool `rtIsTriangleHitBackFace` ()
- static __device__ bool `rtIsTriangleHitFrontFace` ()
- static __device__ void `rtThrow` (unsigned int code)
- static __device__ unsigned int `rtGetExceptionCode` ()
- static __device__ void `rtPrintExceptionDetails` ()

5.24.1 Detailed Description

OptiX Functions designed to operate on device side. Some of them can also be included explicitly in host code if desired

5.24.2 Function Documentation

5.24.2.1 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`, `RException`

5.24.2.2 static __device__ unsigned int rtGetPrimitiveIndex() [inline], [static]

Get the index of the closest hit or currently intersecting primitive.

Description

`rtGetPrimitiveIndex` provides the primitive index similar to what is normally passed to a custom intersection program as an argument. If an primitive-index offset is specified on the geometry (Geometry or GeometryTriangles node), `rtGetPrimitiveIndex` reports the primitive index of the geometry (range [0;N-1] for N primitives) plus the offset. This behavior is equal to what is passed to an intersection program. The `rtGetPrimitiveIndex` semantic is available in any hit, closest hit, and intersection programs.

Return values

<i>unsigned</i>	int index of the primitive
-----------------	----------------------------

History

`rtGetPrimitiveIndex` was introduced in OptiX 6.0.

See also

5.24.2.3 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

void	void return value
------	-------------------

History

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

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

5.24.2.4 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

void	void return value
------	-------------------

History

`rtIgnoreIntersection` was introduced in OptiX 1.0.

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

5.24.2.5 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

in	<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](#)

5.24.2.6 static __device__ bool rtIsTriangleHit() [inline], [static]

Returns if the hit kind of the closest hit or currently intersecting primitive is a builtin triangle.

Description

[rtIsTriangleHit](#) returns true if the intersected primitive is a builtin triangle.

Return values

<i>bool</i>	builtin triangle hit
-------------	----------------------

History

[rtIsTriangleHit](#) was introduced in OptiX 6.0.

See also [rtIsTriangleHitBackFace](#) [rtIsTriangleHitFrontFace](#)

5.24.2.7 static __device__ bool rtIsTriangleHitBackFace() [inline], [static]

Returns if the back face of a builtin triangle was hit.

Description

[rtIsTriangleHitBackFace](#) returns true if the intersected primitive is a builtin triangle and if the back face of that triangle is hit. Returns false otherwise.

Return values

<i>bool</i>	builtin triangle hit back face
-------------	--------------------------------

History

`rtIsTriangleHitFrontFace` was introduced in OptiX 6.0.

See also `rtIsTriangleHit` `rtIsTriangleHitFrontFace`

5.24.2.8 static __device__ bool `rtIsTriangleHitFrontFace()` [inline], [static]

Returns if the front face of a builtin triangle was hit.

Description

`rtIsTriangleHitFrontFace` returns true if the intersected primitive is a builtin triangle and if the front face of that triangle is hit. Returns false otherwise.

Return values

<code>bool</code>	builtin triangle hit front face
-------------------	---------------------------------

History

`rtIsTriangleHitFrontFace` was introduced in OptiX 6.0.

See also `rtIsTriangleHit` `rtIsTriangleHitBackFace`

5.24.2.9 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`.

`rtReportIntersection` 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

<code>in</code>	<code>tmin</code>	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`

5.24.2.10 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` functions 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` functions, `RTexception`

5.24.2.11 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

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

Return values

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

`rtReportIntersection` was introduced in OptiX 1.0.

See also `rtPotentialIntersection`, `rtIgnoreIntersection`

5.24.2.12 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`

5.24.2.13 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 `RT_EXCEPTION_USER_MAX` (`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`, `Rtexception`

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

```
RTvisibilitymask mask = RT_VISIBILITY_ALL,
RTrayflags flags = RT_RAY_FLAG_NONE ) [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`.

The optional *time* argument sets the time of the ray for motion-aware traversal and shading. The ray time is available in user programs as the `rtcurrentTime` semantic variable. If *time* is omitted, then the ray inherits the time of the parent ray that triggered the current program. In a ray generation program where there is no parent ray, the time defaults to 0.0.

The optional visibility *mask* controls intersection against user-configurable groups of objects. Visibility masks of groups and geometries are compared against this mask. Intersections are computed if at least one bit is present in both sets, i.e. if

```
(group_mask & ray_mask) != 0
```

. Note that visibility is currently limited to eight groups, only the lower eight bits of *mask* will be taken into account.

Parameters

in	<i>topNode</i>	Top node object where to start the traversal
in	<i>ray</i>	Ray to be traced
in	<i>time</i>	Time value for the ray
in	<i>prd</i>	Per-ray custom data
in	<i>mask</i>	Visibility mask as described above
in	<i>flags</i>	Ray flags

Return values

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

History

- `rtTrace` was introduced in OptiX 1.0.
- *time* was introduced in OptiX 5.0.
- *mask* and *flags* were introduced in OptiX 6.0.

See also [rtObject](#) [rtcurrentTime](#) [Ray](#) [RTrayflags](#)

5.24.2.15 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

<code>float3</code>	Transformed normal
---------------------	--------------------

History

`rtTransformNormal` was introduced in OptiX 1.0.

See also `rtTransformCreate`, `rtTransformPoint`, `rtTransformVector`

5.24.2.16 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

in	<i>kind</i>	Type of the transform
----	-------------	-----------------------

Parameters

in	<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`

```
5.24.2.17 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

in	<i>kind</i>	Type of the transform
in	<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`

5.25 Texture fetch functions

- `__device__ uint3 optix::rtTexSize(rtTextureId id)`

5.25.1 Detailed Description

5.25.2 Function Documentation

5.25.2.1 `__device__ uint3 optix::rtTexSize(` `rtTextureId id) [inline]`

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

`rtTex1D`, `rtTex2D` and `rtTex3D` fetch the texture referenced by the `id` with texture coordinate `x`, `y` and `z`. The texture sampler `id` can be obtained on the host side using `rtTextureSamplerGetId` function. There are also C++ template and C-style additional declarations for other texture types (`char1`, `uchar1`, `char2`, `uchar2` ...):

To get texture size dimensions `rtTexSize` can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: `rtTex1DFetch`, `rtTex2DFetch` and `rtTex3DFetch`

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: `rtTex2DGather`, `rtTex1DGrad`, `rtTex2DGrad`, `rtTex3DGrad`, `rtTex1DLayeredGrad`, `rtTex2DLayeredGrad`, `rtTex1DLod`, `rtTex2DLod`, `rtTex3DLod`, `rtTex1DLayeredLod`, `rtTex2DLayeredLod`, `rtTex1DLayered`, `rtTex2DLayered`.

And cubeamp textures with `rtTexCubemap`, `rtTexCubemapLod`, `rtTexCubemapLayered` and `rtTexCubemapLayeredLod`.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

`rtTex1D`, `rtTex2D` and `rtTex3D` were introduced in OptiX 3.0.

`rtTexSize`, `rtTex1DFetch`, `rtTex2DFetch`, `rtTex3DFetch`, `rtTex2DGather`, `rtTex1DGrad`, `rtTex2DGrad`, `rtTex3DGrad`, `rtTex1DLayeredGrad`, `rtTex2DLayeredGrad`, `rtTex1DLod`, `rtTex2DLod`, `rtTex3DLod`, `rtTex1DLayeredLod`, `rtTex2DLayeredLod`, `rtTex1DLayered`, `rtTex2DLayered`, `rtTexCubemap`, `rtTexCubemapLod`, `rtTexCubemapLayered` and `rtTexCubemapLayeredLod` were introduced in OptiX 3.9.

See also `rtTextureSamplerGetId`

5.26 rtPrintf functions

- 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)
- 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, typename T13>
 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, T13 arg13)

5.26.1 Detailed Description

5.26.2 Function Documentation

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

Prints text to the standard output.

Description

[rtPrintf functions](#) 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 functions](#) 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 functions](#) invocations will be silently ignored.

History

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

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

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

Prints text to the standard output.

Description

[rtPrintf functions](#) 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 functions](#) 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 functions](#) invocations will be silently ignored.

History

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

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

5.26.2.3 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` functions 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` functions 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` functions invocations will be silently ignored.

History

`rtPrintf` functions was introduced in OptiX 1.0.

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

**5.26.2.4 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` functions 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` functions 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` functions invocations will be silently ignored.

History

`rtPrintf` functions was introduced in OptiX 1.0.

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

**5.26.2.5 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 functions 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 functions 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 functions invocations will be silently ignored.

History

rtPrintf functions was introduced in OptiX 1.0.

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

5.26.2.6 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 functions 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 functions 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 functions invocations will be silently ignored.

History

rtPrintf functions was introduced in OptiX 1.0.

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

5.26.2.7 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 functions](#) 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 functions](#) 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 functions](#) invocations will be silently ignored.

History

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

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

```
5.26.2.8 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 functions](#) 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 functions](#) 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 functions](#) invocations will be silently ignored.

History

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

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

```
5.26.2.9 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` functions 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` functions 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` functions invocations will be silently ignored.

History

`rtPrintf` functions was introduced in OptiX 1.0.

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

```
5.26.2.10 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` functions 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` functions 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` functions invocations will be silently ignored.

History

`rtPrintf` functions was introduced in OptiX 1.0.

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

```
5.26.2.11 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` functions 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` functions 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` functions invocations will be silently ignored.

History

`rtPrintf` functions was introduced in OptiX 1.0.

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

```
5.26.2.12 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](#) functions 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](#) functions 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](#) functions invocations will be silently ignored.

History

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

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

```
5.26.2.13 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` functions 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` functions 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` functions invocations will be silently ignored.

History

`rtPrintf` functions was introduced in OptiX 1.0.

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

```
5.26.2.14 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 , typename T13 > 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,
T13 arg13 ) [inline], [static]
```

Prints text to the standard output.

Description

`rtPrintf` functions 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` functions 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](#) functions invocations will be silently ignored.

History

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

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

5.27 OptiXpp wrapper

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::GeometryTrianglesObj`
- class `optix::MaterialObj`
- class `optix::TextureSamplerObj`
- class `optix::BufferObj`
- class `optix::PostprocessingStageObj`
- class `optix::CommandListObj`

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

5.27.1 Detailed Description

5.27.2 Typedef Documentation

5.27.2.1 **typedef Handle<AccelerationObj> optix::Acceleration**

Use this to manipulate RTacceleration objects.

5.27.2.2 **typedef Handle<BufferObj> optix::Buffer**

Use this to manipulate RTbuffer objects.

5.27.2.3 **typedef Handle<CommandListObj> optix::CommandList**

Use this to manipulate RTcommandlist objects.

5.27.2.4 **typedef Handle<ContextObj> optix::Context**

Use this to manipulate RTcontext objects.

5.27.2.5 **typedef Handle<GeometryObj> optix::Geometry**

Use this to manipulate RTgeometry objects.

5.27.2.6 **typedef Handle<GeometryGroupObj> optix::GeometryGroup**

Use this to manipulate RTgeometrygroup objects.

5.27.2.7 **typedef Handle<GeometryInstanceObj> optix::GeometryInstance**

Use this to manipulate RTgeometryinstance objects.

5.27.2.8 **typedef Handle<GeometryTrianglesObj> optix::GeometryTriangles**

Use this to manipulate RTgeometrytriangles objects.

5.27.2.9 **typedef Handle<GroupObj> optix::Group**

Use this to manipulate RTgroup objects.

5.27.2.10 **typedef Handle<MaterialObj> optix::Material**

Use this to manipulate RTmaterial objects.

5.27.2.11 **typedef Handle<PostprocessingStageObj> optix::PostprocessingStage**

Use this to manipulate RTpostprocessingstage objects.

5.27.2.12 **typedef Handle<ProgramObj> optix::Program**

Use this to manipulate RTprogram objects.

5.27.2.13 `typedef Handle<SelectorObj> optix::Selector`

Use this to manipulate RTselector objects.

5.27.2.14 `typedef Handle<TextureSamplerObj> optix::TextureSampler`

Use this to manipulate RTtexturesampler objects.

5.27.2.15 `typedef Handle<TransformObj> optix::Transform`

Use this to manipulate RTtransform objects.

5.27.2.16 `typedef Handle<VariableObj> optix::Variable`

Use this to manipulate RTvariable objects.

5.28 rtu API

Modules

- rtu Traversal API

Functions

- RTresult RTAPI rtuNameForType (RTobjecttype type, char *buffer, RTsize bufferSize)
- RTresult RTAPI rtuGetSizeForRTformat (RTformat format, size_t *size)
- RTresult RTAPI rtuCUDACompileString (const char *source, const char **preprocessorArguments, unsigned int numPreprocessorArguments, RTsize *resultSize, RTsize *errorSize)
- RTresult RTAPI rtuCUDACompileFile (const char *filename, const char **preprocessorArguments, unsigned int numPreprocessorArguments, RTsize *resultSize, RTsize *errorSize)
- RTresult RTAPI rtuCUDAGetCompileResult (char *result, char *error)
- static RTresult rtuGroupAddChild (RTgroup group, RTobject child, unsigned int *index)
- static RTresult rtuSelectorAddChild (RTselector selector, RTobject child, unsigned int *index)
- static RTresult rtuGeometryGroupAddChild (RTgeometrygroup geometrygroup, RTgeometryinstance child, unsigned int *index)
- static RTresult rtuTransformSetChild (RTtransform transform, RTobject child)
- static RTresult rtuTransformGetChild (RTtransform transform, RTobject *type)
- static RTresult rtuTransformGetChildType (RTtransform transform, RTobjecttype *type)
- static RTresult rtuGroupRemoveChild (RTgroup group, RTobject child)
- static RTresult rtuSelectorRemoveChild (RTselector selector, RTobject child)
- static RTresult rtuGeometryGroupRemoveChild (RTgeometrygroup geometrygroup, RTgeometryinstance child)
- static RTresult rtuGroupRemoveChildByIndex (RTgroup group, unsigned int index)
- static RTresult rtuSelectorRemoveChildByIndex (RTselector selector, unsigned int index)
- static RTresult rtuGeometryGroupRemoveChildByIndex (RTgeometrygroup geometrygroup, unsigned int index)
- static RTresult rtuGroupGetChildIndex (RTgroup group, RTobject child, unsigned int *index)
- static RTresult rtuSelectorGetChildIndex (RTselector selector, RTobject child, unsigned int *index)
- static RTresult rtuGeometryGroupGetChildIndex (RTgeometrygroup geometrygroup, RTgeometryinstance child, unsigned int *index)

5.28.1 Detailed Description

The rtu API provides a simple interface for intersecting a set of rays against a set of triangles. It has been superseded by OptiX Prime.

5.28.2 Function Documentation

5.28.2.1 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>preprocessorArguments</i>	list of preprocessor arguments
in	<i>numPreprocessorArguments</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
-----------------	-------------

5.28.2.2 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>preprocessorArguments</i>	list of preprocessor arguments
in	<i>numPreprocessorArguments</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
-----------------	-------------

5.28.2.3 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
-----------------	-------------

5.28.2.4 static RTResult rtuGeometryGroupAddChild (

```
RTgeometrygroup geometrygroup,
RTgeometryinstance child,
unsigned int * index ) [inline], [static]
```

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.

5.28.2.5 static RTResult rtuGeometryGroupGetChildIndex (

```
RTgeometrygroup geometrygroup,
RTgeometryinstance child,
unsigned int * index ) [inline], [static]
```

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.

5.28.2.6 static RTResult rtuGeometryGroupRemoveChild (

```
RTgeometrygroup geometrygroup,
RTgeometryinstance child ) [inline], [static]
```

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.

5.28.2.7 static RTResult rtuGeometryGroupRemoveChildByIndex (

```
RTgeometrygroup geometrygroup,
```

unsigned int *index*) [inline], [static]

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.

5.28.2.8 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

RTResult	Return code
-----------------	-------------

5.28.2.9 static RTresult rtuGroupAddChild (

RTgroup *group*,
RTobject *child*,
unsigned int * *index*) [inline], [static]

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.

5.28.2.10 static RTresult rtuGroupGetChildIndex (

RTgroup *group*,
RTobject *child*,
unsigned int * *index*) [inline], [static]

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.

5.28.2.11 static RTresult rtuGroupRemoveChild (

RTgroup *group*,
RTobject *child*) [inline], [static]

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.

5.28.2.12 static RTresult rtuGroupRemoveChildByIndex (

RTgroup *group*,

unsigned int *index*) [inline], [static]

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.

5.28.2.13 RTResult RTAPI rtuNameForType (

RTobjecttype *type*,

char * *buffer*,

RTsize *bufferSize*)

Get the name string of a given type.

See [RTobjecttype](#) 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
-----------------	-------------

5.28.2.14 static RTresult rtuSelectorAddChild (

RTselector *selector*,

RTobject *child*,

unsigned int * *index*) [inline], [static]

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.

5.28.2.15 static RTResult rtuSelectorGetChildIndex (

RTselector *selector*,

RTobject *child*,

unsigned int * *index*) [inline], [static]

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.

5.28.2.16 static RTresult rtuSelectorRemoveChild (
 RTselector *selector*,
 RTobject *child*) [inline], [static]

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.

5.28.2.17 static RTresult rtuSelectorRemoveChildByIndex (
 RTselector *selector*,
 unsigned int *index*) [inline], [static]

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.

5.28.2.18 static RTresult rtuTransformGetChild (
 RTtransform *transform*,
 RTobject * *type*) [inline], [static]

Wrap rtTransformGetChild and rtTransformGetChildType in order to provide a type-safe version for C++.

5.28.2.19 static RTresult rtuTransformGetChildType (
 RTtransform *transform*,
 RTobjecttype * *type*) [inline], [static]

Wrap rtTransformGetChild and rtTransformGetChildType in order to provide a type-safe version for C++.

5.28.2.20 static RTresult rtuTransformSetChild (
 RTtransform *transform*,
 RTobject *child*) [inline], [static]

Wrap rtTransformSetChild in order to provide a type-safe version for C++.

5.29 rtu Traversal API

5.30 OptiX Prime API Reference

Modules

- Context
- Buffer descriptor
- Model
- Query
- Miscellaneous functions
- OptiX Prime++ wrapper

5.30.1 Detailed Description

5.31 Context

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)

5.31.1 Detailed Description

5.31.2 Function Documentation

5.31.2.1 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 the fastest available CUDA device, but note that specific devices can be selected using `rtpContextSetCudaDeviceNumbers`. The fastest device will be set as the current device when the function returns. If no CUDA device features compute capability 3.0 or greater, the context creation will fail unless `RTP_CONTEXT_TYPE_CPU` was specified.

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
}

```

5.31.2.2 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

5.31.2.3 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](#)

5.31.2.4 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`

5.31.2.5 RTPResult RTPAPI rtpContextSetCudaDeviceNumbers (

```
RTPcontext context,
unsigned deviceCount,
const unsigned * deviceNumbers )
```

Sets the CUDA devices used by a context.

The fastest device provided in *deviceNumbers* will be used as the *primary device*. Acceleration structures will be built on that primary device and copied to the others. All devices must be of compute capability 3.0 or greater. Note that this distribution can be rather costly if the rays are stored in device memory though. For maximum efficiency it is recommended to only ever select one device per context. 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==0</i>

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

5.32 Buffer descriptor

Functions

- 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)

5.32.1 Detailed Description

5.32.2 Function Documentation

5.32.2.1 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);
```

5.32.2.2 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	<i>desc</i>	Buffer descriptor
----	-------------	-------------------

Return values

Relevant return values:

- RTP_SUCCESS
- RTP_ERROR_INVALID_VALUE
- RTP_ERROR_UNKNOWN

5.32.2.3 RTPResult RTPAPI rtpBufferDescGetContext (RTPbufferdesc desc, RTPcontext * context)

Gets the context object associated with the provided buffer descriptor.

Parameters

in	<i>desc</i>	Buffer descriptor
out	<i>context</i>	Returned context

Return values

Relevant return values:

- RTP_SUCCESS
- RTP_ERROR_INVALID_VALUE
- RTP_ERROR_UNKNOWN

5.32.2.4 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	<i>desc</i>	Buffer descriptor
in	<i>deviceNumber</i>	CUDA device number

Return values

Relevant return values:

- `RTP_SUCCESS`
- `RTP_ERROR_INVALID_VALUE`
- `RTP_ERROR_UNKNOWN`

5.32.2.5 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`

5.32.2.6 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	<code>desc</code>	Buffer descriptor
in	<code>strideBytes</code>	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));
```

5.33 Model

Functions

- 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)

5.33.1 Detailed Description

5.33.2 Function Documentation

5.33.2.1 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

5.33.2.2 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

5.33.2.3 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

in	<i>model</i>	Model
----	--------------	-------

Return values

Relevant return values:

- RTP_SUCCESS
- RTP_ERROR_INVALID_VALUE
- RTP_ERROR_UNKNOWN

5.33.2.4 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

in	<i>model</i>	Model
----	--------------	-------

Return values

Relevant return values:

- RTP_SUCCESS
- RTP_ERROR_INVALID_VALUE
- RTP_ERROR_UNKNOWN

5.33.2.5 RTPResult RTPAPI rtpModelGetContext (

RTPmodel *model*,
RTPcontext * *context*)

Gets the context object associated with the model.

Parameters

in	<i>model</i>	Model 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

5.33.2.6 RTPResult RTPAPI rtpModelGetFinished (

RTPmodel *model*,
int * *isFinished*)

Polls the status of a model update.

Parameters

in	<i>model</i>	Model
out	<i>isFinished</i>	Returns finished status

Return values

Relevant return values:

- RTP_SUCCESS
- RTP_ERROR_INVALID_VALUE

- `RTP_ERROR_UNKNOWN`

5.33.2.7 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

<code>in</code>	<i>model_api</i>	Model
<code>in</code>	<i>param</i>	Builder parameter to set
<code>in</code>	<i>size</i>	Size in bytes of the parameter being set
<code>in</code>	<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`

5.33.2.8 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`

5.33.2.9 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

5.33.2.10 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 <code>RTPmodelhint</code>

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
```

5.34 Query

Functions

- 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)

5.34.1 Detailed Description

5.34.2 Function Documentation

5.34.2.1 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

5.34.2.2 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

5.34.2.3 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(model1, RTP_QUERY_TYPE_CLOSEST, &query);  
rtpQuerySetRays(query, raysBD);  
rtpQuerySetHits(hits, hitsBD);  
rtpQueryExecute(query, 0);  
// safe to modify ray buffer and process hits
```

5.34.2.4 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

5.34.2.5 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

5.34.2.6 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

5.34.2.7 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

5.34.2.8 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

5.34.2.9 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

5.35 Miscellaneous functions

Functions

- 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)

5.35.1 Detailed Description

5.35.2 Function Documentation

5.35.2.1 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](#)

5.35.2.2 RTPResult RTPAPI rtpGetVersion (

unsigned * version)

Gets OptiX Prime version number.

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 4.1.2 it would return 40102.

Parameters

out	<i>version</i>	Returned version
-----	----------------	------------------

Return values

Relevant return values:

- RTP_SUCCESS
- RTP_ERROR_INVALID_VALUE

5.35.2.3 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

5.35.2.4 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

5.35.2.5 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	buffer	Buffer on the host
----	--------	--------------------

Return values

Relevant return values:

- `RTP_SUCCESS`
- `RTP_ERROR_INVALID_VALUE`

5.36 OptiX Prime++ wrapper

Classes

- class `optix::prime::ContextObj`
- class `optix::prime::BufferDescObj`
- class `optix::prime::ModelObj`
- class `optix::prime::QueryObj`
- class `optix::prime::Exception`

- typedef Handle< BufferDescObj > `optix::prime::BufferDesc`
- typedef Handle< ContextObj > `optix::prime::Context`
- typedef Handle< ModelObj > `optix::prime::Model`
- typedef Handle< QueryObj > `optix::prime::Query`

5.36.1 Detailed Description

5.36.2 Typedef Documentation

5.36.2.1 `typedef Handle<BufferDescObj> optix::prime::BufferDesc`

Use this to manipulate RTPbufferdesc objects.

5.36.2.2 `typedef Handle<ContextObj> optix::prime::Context`

Use this to manipulate RTPcontext objects.

5.36.2.3 `typedef Handle<ModelObj> optix::prime::Model`

Use this to manipulate RTPmodel objects.

5.36.2.4 `typedef Handle<QueryObj> optix::prime::Query`

Use this to manipulate RTPquery objects.

5.37 OptiX Interoperability Types

Modules

- OpenGL Texture Formats
- DXGI Texture Formats

5.37.1 Detailed Description

This section lists OpenGL and Direct3D texture formats that are currently supported for interoperability with OptiX.

5.38 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

5.39 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

6 Namespace Documentation

6.1 optix Namespace Reference

Namespaces

- `detail`
- `prime`

Classes

- `struct VectorTypes`
- `struct VectorTypes< int, 1 >`
- `struct VectorTypes< int, 2 >`
- `struct VectorTypes< int, 3 >`
- `struct VectorTypes< int, 4 >`
- `struct VectorTypes< unsigned int, 1 >`
- `struct VectorTypes< unsigned int, 2 >`
- `struct VectorTypes< unsigned int, 3 >`
- `struct VectorTypes< unsigned int, 4 >`
- `struct VectorTypes< float, 1 >`
- `struct VectorTypes< float, 2 >`
- `struct VectorTypes< float, 3 >`
- `struct VectorTypes< float, 4 >`
- `struct bufferId`
- `struct buffer`
- `class callableProgramId`
- `class markedCallableProgramId`
- `class boundCallableProgramId`
- `class PtxEncryption`
- `class Handle`
- `class Exception`
- `class APIObj`
- `class DestroyableObj`
- `class ScopedObj`
- `class VariableObj`
- `class ContextObj`
- `class ProgramObj`
- `class GroupObj`
- `class GeometryGroupObj`
- `class TransformObj`
- `class SelectorObj`
- `class AccelerationObj`
- `class GeometryInstanceObj`
- `class GeometryObj`
- `class GeometryTrianglesObj`

- class [MaterialObj](#)
- class [TextureSamplerObj](#)
- class [BufferObj](#)
- class [PostprocessingStageObj](#)
- class [CommandListObj](#)
- class [Aabb](#)
- struct [Onb](#)
- struct [VectorDim](#)
- struct [VectorDim< 2 >](#)
- struct [VectorDim< 3 >](#)
- struct [VectorDim< 4 >](#)
- class [Matrix](#)
- class [Quaternion](#)

Typedefs

- [typedef size_t optix_size_t](#)
- [typedef int rtTextureId](#)
- [typedef unsigned int uint](#)
- [typedef unsigned short ushort](#)
- [typedef Matrix< 2, 2 > Matrix2x2](#)
- [typedef Matrix< 2, 3 > Matrix2x3](#)
- [typedef Matrix< 2, 4 > Matrix2x4](#)
- [typedef Matrix< 3, 2 > Matrix3x2](#)
- [typedef Matrix< 3, 3 > Matrix3x3](#)
- [typedef Matrix< 3, 4 > Matrix3x4](#)
- [typedef Matrix< 4, 2 > Matrix4x2](#)
- [typedef Matrix< 4, 3 > Matrix4x3](#)
- [typedef Matrix< 4, 4 > Matrix4x4](#)

- [typedef Handle< AccelerationObj > Acceleration](#)
- [typedef Handle< BufferObj > Buffer](#)
- [typedef Handle< ContextObj > Context](#)
- [typedef Handle< GeometryObj > Geometry](#)
- [typedef Handle< GeometryTrianglesObj > GeometryTriangles](#)
- [typedef Handle< GeometryGroupObj > GeometryGroup](#)
- [typedef Handle< GeometryInstanceObj > GeometryInstance](#)
- [typedef Handle< GroupObj > Group](#)
- [typedef Handle< MaterialObj > Material](#)
- [typedef Handle< ProgramObj > Program](#)
- [typedef Handle< SelectorObj > Selector](#)
- [typedef Handle< TextureSamplerObj > TextureSampler](#)
- [typedef Handle< TransformObj > Transform](#)
- [typedef Handle< VariableObj > Variable](#)

- `typedef Handle< PostprocessingStageObj > PostprocessingStage`
- `typedef Handle< CommandListObj > CommandList`

Enumerations

- `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 }`
- `enum ObjectStorageType {
 OBJECT_STORAGE_CONSTANT,
 OBJECT_STORAGE_SHARED,
 OBJECT_STORAGE_GLOBAL,
 OBJECT_STORAGE_LINEAR_TEXTURE,
 OBJECT_STORAGE_BLOCKED_TEXTURE }`

Functions

- `void rt_undefined_use (int)`
- `void rt_undefined_use64 (int)`
- `static __forceinline__
 __device__ uint3 rt_texture_get_size_id (int tex)`
- `static __forceinline__
 __device__ float4 rt_texture_get_gather_id (int tex, float x, float y, int comp)`
- `static __forceinline__
 __device__ float4 rt_texture_get_base_id (int tex, int dim, float x, float y, float z, int layer)`
- `static __forceinline__
 __device__ float4 rt_texture_get_level_id (int tex, int dim, float x, float y, float z, int layer, float level)`
- `static __forceinline__
 __device__ float4 rt_texture_get_grad_id (int tex, int dim, float x, float y, float z, int layer, float dPdx_x, float dPdx_y, float dPdx_z, float dPdy_x, float dPdy_y, float dPdy_z)`
- `static __forceinline__
 __device__ float4 rt_texture_get_f_id (int tex, int dim, float x, float y, float z, float w)`
- `static __forceinline__
 __device__ int4 rt_texture_get_i_id (int tex, int dim, float x, float y, float z, float w)`
- `static __forceinline__
 __device__ uint4 rt_texture_get_u_id (int tex, int dim, float x, float y, float z, float w)`
- `static __forceinline__
 __device__ float4 rt_texture_get_fetch_id (int tex, int dim, int x, int y, int z, int w)`

- static __forceinline__
 __device__ void * rt_buffer_get (void *buffer, unsigned int dim, unsigned int element_size, size_t i0_in, size_t i1_in, size_t i2_in, size_t i3_in)
- static __forceinline__
 __device__ void * rt_buffer_get_id (int id, unsigned int dim, unsigned int element_size, size_t i0_in, size_t i1_in, size_t i2_in, size_t i3_in)
- static __forceinline__
 __device__ size_t4 rt_buffer_get_size (const void *buffer, unsigned int dim, unsigned int element_size)
- static __forceinline__
 __device__ size_t4 rt_buffer_get_size_id (int id, unsigned int dim, unsigned int element_size)
- static __forceinline__
 __device__ void * rt_callable_program_from_id (int id, const char *csld=0)
- static __forceinline__
 __device__ void rt_trace (unsigned int group, float3 origin, float3 direction, unsigned int ray_type, float tmin, float tmax, RTvisibilitymask mask, RTrayflags flags, void *prd, unsigned int prd_size)
- static __forceinline__
 __device__ void rt_trace_with_time (unsigned int group, float3 origin, float3 direction, unsigned int ray_type, float tmin, float tmax, float time, RTvisibilitymask mask, RTrayflags flags, void *prd, unsigned int prd_size)
- static __forceinline__
 __device__ bool rt_potential_intersection (float t)
- static __forceinline__
 __device__ bool rt_report_intersection (unsigned int matlIndex)
- static __forceinline__
 __device__ void rt_ignore_intersection ()
- static __forceinline__
 __device__ void rt_terminate_ray ()
- static __forceinline__
 __device__ void rt_intersect_child (unsigned int index)
- static __forceinline__
 __device__ float3 rt_transform_point (RTtransformkind kind, const float3 &p)
- static __forceinline__
 __device__ float3 rt_transform_vector (RTtransformkind kind, const float3 &v)
- static __forceinline__
 __device__ float3 rt_transform_normal (RTtransformkind kind, const float3 &n)
- static __forceinline__
 __device__ void rt_get_transform (RTtransformkind kind, float matrix[16])
- static __forceinline__
 __device__ unsigned int rt_get_primitive_index ()
- static __forceinline__
 __device__ bool rt_is_triangle_hit ()
- static __forceinline__
 __device__ bool rt_is_triangle_hit_back_face ()
- static __forceinline__
 __device__ bool rt_is_triangle_hit_front_face ()
- static __forceinline__
 __device__ void rt_throw (unsigned int code)

- static __forceinline__
 __device__ unsigned int **rt_get_exception_code** ()
- static __forceinline__
 __device__ int **rt_print_active** ()
- static __forceinline__
 __device__ float2 **rt_get_triangle_barycentrics** ()
- __device__ int4 **float4AsInt4** (float4 f4)
- __device__ uint4 **float4AsUInt4** (float4 f4)
- template<typename ReturnT >
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_NO_ARG (ReturnT)
- template<typename ReturnT , typename Arg0T >
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T)
- template<typename ReturnT , typename Arg0T , typename Arg1T >
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T)
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T >
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T)
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T >
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T)
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T >
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T)
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T >
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T)
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T >
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T)
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T , typename Arg7T >
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T)
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T , typename Arg7T , typename Arg8T >
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T)
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T , typename Arg7T , typename Arg8T , typename Arg9T >
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T)
- template<typename ReturnT >
 class **callableProgramId**< ReturnT()> **RT_INTERNAL_CALLABLE_PROGRAM_DEFS** ()
- template<typename ReturnT , typename Arg0T >

```

class callableProgramId
< ReturnT(Arg0T)> RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()
• template<typename ReturnT , typename Arg0T , typename Arg1T >
class callableProgramId
< ReturnT(Arg0T, Arg1T)> RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()
• template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T >
class callableProgramId
< ReturnT(Arg0T, Arg1T, Arg2T)> RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()
• template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T >
class callableProgramId
< ReturnT(Arg0T, Arg1T, Arg2T,
Arg3T)> 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)> 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)> 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)> 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)> 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)> 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)> RT_INTERNAL_CALLABLE_PROGRAM_DEFS ()
• OPTIXU_INLINE float fminf (const float a, const float b)
• OPTIXU_INLINE float fmaxf (const float a, const float b)
• OPTIXU_INLINE float copysignf (const float dst, const float src)

```

- `OPTIXU_INLINE int max (int a, int b)`
- `OPTIXU_INLINE int min (int a, int b)`
- `OPTIXU_INLINE long long max (long long a, long long b)`
- `OPTIXU_INLINE long long min (long long a, long long b)`
- `OPTIXU_INLINE unsigned int max (unsigned int a, unsigned int b)`
- `OPTIXU_INLINE unsigned int min (unsigned int a, unsigned int b)`
- `OPTIXU_INLINE unsigned long long max (unsigned long long a, unsigned long long b)`
- `OPTIXU_INLINE unsigned long long min (unsigned long long a, unsigned long long b)`
- `OPTIXU_INLINE int float_as_int (const float f)`
- `OPTIXU_INLINE float int_as_float (int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE float lerp (const float a, const float b, const float t)`
- `OPTIXU_INLINE RT_HOSTDEVICE float bilerp (const float x00, const float x10, const float x01, const float x11, const float u, const float v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float clamp (const float f, const float a, const float b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float getByIndex (const float1 &v, int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (float1 &v, int i, float x)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator- (const float2 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 lerp (const float2 &a, const float2 &b, const float t)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 bilerp (const float2 &x00, const float2 &x10, const float2 &x01, const float2 &x11, const float u, const float v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float dot (const float2 &a, const float2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float length (const float2 &v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 normalize (const float2 &v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 floor (const float2 &v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 reflect (const float2 &i, const float2 &n)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 faceforward (const float2 &n, const float2 &i, const float2 &nref)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 expf (const float2 &v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float getByIndex (const float2 &v, int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (float2 &v, int i, float x)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator- (const float3 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 lerp (const float3 &a, const float3 &b, const float t)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 bilerp (const float3 &x00, const float3 &x10, const float3 &x01, const float3 &x11, const float u, const float v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float dot (const float3 &a, const float3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 cross (const float3 &a, const float3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float length (const float3 &v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 normalize (const float3 &v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 floor (const float3 &v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 reflect (const float3 &i, const float3 &n)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 faceforward (const float3 &n, const float3 &i, const float3 &nref)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 expf (const float3 &v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float getByIndex (const float3 &v, int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (float3 &v, int i, float x)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 operator- (const float4 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE float4 lerp (const float4 &a, const float4 &b, const float t)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 bilerp (const float4 &x00, const float4 &x10, const float4 &x01, const float4 &x11, const float u, const float v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float dot (const float4 &a, const float4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float length (const float4 &r)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 normalize (const float4 &v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 floor (const float4 &v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 reflect (const float4 &i, const float4 &n)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 faceforward (const float4 &n, const float4 &i, const float4 &nref)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 expf (const float4 &v)`
- `OPTIXU_INLINE RT_HOSTDEVICE float getByIndex (const float4 &v, int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (float4 &v, int i, float x)`
- `OPTIXU_INLINE RT_HOSTDEVICE int clamp (const int f, const int a, const int b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int getByIndex (const int1 &v, int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (int1 &v, int i, int x)`
- `OPTIXU_INLINE RT_HOSTDEVICE int2 operator- (const int2 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE int2 min (const int2 &a, const int2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int2 max (const int2 &a, const int2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int getByIndex (const int2 &v, int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (int2 &v, int i, int x)`
- `OPTIXU_INLINE RT_HOSTDEVICE int3 operator- (const int3 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE int3 min (const int3 &a, const int3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int3 max (const int3 &a, const int3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int getByIndex (const int3 &v, int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (int3 &v, int i, int x)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 operator- (const int4 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 min (const int4 &a, const int4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 max (const int4 &a, const int4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int getByIndex (const int4 &v, int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (int4 &v, int i, int x)`
- `OPTIXU_INLINE RT_HOSTDEVICE`
 `unsigned int clamp (const unsigned int f, const unsigned int a, const unsigned int b)`
- `OPTIXU_INLINE RT_HOSTDEVICE`
 `unsigned int getByIndex (const uint1 &v, unsigned int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (uint1 &v, int i, unsigned int x)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint2 min (const uint2 &a, const uint2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint2 max (const uint2 &a, const uint2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE`
 `unsigned int getByIndex (const uint2 &v, unsigned int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (uint2 &v, int i, unsigned int x)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint3 min (const uint3 &a, const uint3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint3 max (const uint3 &a, const uint3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE`
 `unsigned int getByIndex (const uint3 &v, unsigned int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (uint3 &v, int i, unsigned int x)`

- **OPTIXU_INLINE RT_HOSTDEVICE**
 unsigned int **getByIndex** (const uint4 &v, unsigned int i)
- **OPTIXU_INLINE RT_HOSTDEVICE** void **setByIndex** (uint4 &v, int i, unsigned int x)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 long long **clamp** (const long long f, const long long a, const long long b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 long long **getByIndex** (const longlong1 &v, int i)
- **OPTIXU_INLINE RT_HOSTDEVICE** void **setByIndex** (longlong1 &v, int i, long long x)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 longlong2 **operator-** (const longlong2 &a)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 longlong2 **min** (const longlong2 &a, const longlong2 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 longlong2 **max** (const longlong2 &a, const longlong2 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 long long **getByIndex** (const longlong2 &v, int i)
- **OPTIXU_INLINE RT_HOSTDEVICE** void **setByIndex** (longlong2 &v, int i, long long x)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 longlong3 **operator-** (const longlong3 &a)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 longlong3 **min** (const longlong3 &a, const longlong3 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 longlong3 **max** (const longlong3 &a, const longlong3 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 long long **getByIndex** (const longlong3 &v, int i)
- **OPTIXU_INLINE RT_HOSTDEVICE** void **setByIndex** (longlong3 &v, int i, int x)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 longlong4 **operator-** (const longlong4 &a)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 longlong4 **min** (const longlong4 &a, const longlong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 longlong4 **max** (const longlong4 &a, const longlong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 long long **getByIndex** (const longlong4 &v, int i)
- **OPTIXU_INLINE RT_HOSTDEVICE** void **setByIndex** (longlong4 &v, int i, long long x)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 unsigned long long **clamp** (const unsigned long long f, const unsigned long long a, const unsigned long long b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 unsigned long long **getByIndex** (const ulonglong1 &v, unsigned int i)
- **OPTIXU_INLINE RT_HOSTDEVICE** void **setByIndex** (ulonglong1 &v, int i, unsigned long long x)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 ulonglong2 **min** (const ulonglong2 &a, const ulonglong2 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 ulonglong2 **max** (const ulonglong2 &a, const ulonglong2 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 unsigned long long **getByIndex** (const ulonglong2 &v, unsigned int i)

- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (ulonglong2 &v, int i, unsigned long long x)`
- `OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 min (const ulonglong3 &a, const ulonglong3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 max (const ulonglong3 &a, const ulonglong3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE unsigned long long getByIndex (const ulonglong3 &v, unsigned int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (ulonglong3 &v, int i, unsigned long long x)`
- `OPTIXU_INLINE RT_HOSTDEVICE unsigned long long getByIndex (const ulonglong4 &v, unsigned int i)`
- `OPTIXU_INLINE RT_HOSTDEVICE void setByIndex (ulonglong4 &v, int i, unsigned long long x)`
- `OPTIXU_INLINE RT_HOSTDEVICE float smoothstep (const float edge0, const float edge1, const float x)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 temperature (const float t)`
- `OPTIXU_INLINE RT_HOSTDEVICE bool 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 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 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 refract (float3 &r, const float3 &i, const float3 &n, const float ior)`
- `OPTIXU_INLINE RT_HOSTDEVICE float 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 fresnel_schlick (const float cos_theta, const float exponent, const float3 &minimum, const float3 &maximum)`
- `OPTIXU_INLINE RT_HOSTDEVICE float luminance (const float3 &rgb)`
- `OPTIXU_INLINE RT_HOSTDEVICE float luminanceCIE (const float3 &rgb)`
- `OPTIXU_INLINE RT_HOSTDEVICE void cosine_sample_hemisphere (const float u1, const float u2, float3 &p)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 square_to_disk (const float2 &sample)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 cart_to_pol (const float3 &v)`
- `std::ostream & operator<< (std::ostream &os, const optix::Aabb &aabb)`
- template<unsigned int M>
- `OPTIXU_INLINE RT_HOSTDEVICE Matrix< M, M > & operator*= (Matrix< M, M > &m1, const Matrix< M, M > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE RT_HOSTDEVICE bool operator== (const Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE RT_HOSTDEVICE bool operator!= (const Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE RT_HOSTDEVICE Matrix< M, N > & operator-= (Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE RT_HOSTDEVICE Matrix< M, N > & operator+= (Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE RT_HOSTDEVICE Matrix< M, N > & operator*= (Matrix< M, N > &m1, float f)`

- **RT_MAT_DECL OPTIXU_INLINE**
`RT_HOSTDEVICE Matrix< M, N > & operator/= (Matrix< M, N > &m1, float f)`
- **RT_MAT_DECL OPTIXU_INLINE**
`RT_HOSTDEVICE Matrix< M, N > operator- (const Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- **RT_MAT_DECL OPTIXU_INLINE**
`RT_HOSTDEVICE Matrix< M, N > operator+ (const Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- **RT_MAT_DECL OPTIXU_INLINE**
`RT_HOSTDEVICE Matrix< M, N > operator/ (const Matrix< M, N > &m, float f)`
- **RT_MAT_DECL OPTIXU_INLINE**
`RT_HOSTDEVICE Matrix< M, N > operator* (const Matrix< M, N > &m, float f)`
- **RT_MAT_DECL OPTIXU_INLINE**
`RT_HOSTDEVICE Matrix< M, N > operator* (float f, const Matrix< M, N > &m)`
- **RT_MAT_DECL OPTIXU_INLINE**
`RT_HOSTDEVICE Matrix< M, N > ::floatM operator* (const Matrix< M, N > &m, const typename Matrix< M, N >::floatN &v)`
- **RT_MAT_DECL OPTIXU_INLINE**
`RT_HOSTDEVICE Matrix< M, N > ::floatN 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 > operator* (const Matrix< M, N > &m1, const Matrix< N, R > &m2)`
- template<unsigned int N>
OPTIXU_INLINE RT_HOSTDEVICE float2 operator* (const Matrix< 2, N > &m, const typename Matrix< 2, N >::floatN &vec)
- template<unsigned int N>
OPTIXU_INLINE RT_HOSTDEVICE float3 operator* (const Matrix< 3, N > &m, const typename Matrix< 3, N >::floatN &vec)
- template<unsigned int N>
OPTIXU_INLINE RT_HOSTDEVICE float4 operator* (const Matrix< 4, N > &m, const typename Matrix< 4, N >::floatN &vec)
- **OPTIXU_INLINE RT_HOSTDEVICE** float4 operator* (const Matrix< 4, 4 > &m, const float4 &vec)
- template<unsigned int M, unsigned int N, unsigned int R>
RT_HOSTDEVICE Matrix< M, R > operator* (const Matrix< M, N > &m1, const Matrix< N, R > &m2)
- template<unsigned int M>
RT_HOSTDEVICE Matrix< M, M > & operator*= (Matrix< M, M > &m1, const Matrix< M, M > &m2)
- **OPTIXU_INLINE RT_HOSTDEVICE**
`Matrix< 3, 3 > make_matrix3x3 (const Matrix< 4, 4 > &matrix)`
- **OPTIXU_INLINE RT_HOSTDEVICE** float3 operator* (const Quaternion &quat, const float3 &v)
- **OPTIXU_INLINE RT_HOSTDEVICE** float4 operator* (const Quaternion &quat, const float4 &v)
- **OPTIXU_INLINE RT_HOSTDEVICE**
`Quaternion nlerp (const Quaternion &quat0, const Quaternion &quat1, float t)`
- **OPTIXU_INLINE RT_HOSTDEVICE** float2 make_float2 (const float s)

- `OPTIXU_INLINE RT_HOSTDEVICE float2 make_float2 (const int2 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 make_float2 (const uint2 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE float2 fminf (const float2 &a, const float2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float fminf (const float2 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE float2 fmaxf (const float2 &a, const float2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float fmaxf (const float2 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator+ (const float2 &a, const float2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator+ (const float2 &a, const float b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator+ (const float a, const float2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator+= (float2 &a, const float2 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator- (const float2 &a, const float2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator- (const float2 &a, const float b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator- (const float a, const float2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator-= (float2 &a, const float2 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator* (const float2 &a, const float2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator* (const float2 &a, const float s)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator* (const float s, const float2 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator*= (float2 &a, const float2 &s)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator*= (float2 &a, const float s)`

- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator/ (const float2 &a, const float2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator/ (const float2 &a, const float s)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 operator/ (const float s, const float2 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator/= (float2 &a, const float s)`

- `OPTIXU_INLINE RT_HOSTDEVICE float2 clamp (const float2 &v, const float a, const float b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 clamp (const float2 &v, const float2 &a, const float2 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE float3 make_float3 (const float s)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 make_float3 (const float2 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 make_float3 (const int3 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 make_float3 (const uint3 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE float3 fminf (const float3 &a, const float3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float fminf (const float3 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE float3 fmaxf (const float3 &a, const float3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float fmaxf (const float3 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator+ (const float3 &a, const float3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator+ (const float3 &a, const float b)`

- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator+ (const float a, const float3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator+= (float3 &a, const float3 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator- (const float3 &a, const float3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator- (const float3 &a, const float b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator- (const float a, const float3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator-= (float3 &a, const float3 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator* (const float3 &a, const float3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator* (const float3 &a, const float s)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator* (const float s, const float3 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator*= (float3 &a, const float3 &s)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator*= (float3 &a, const float s)`

- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator/ (const float3 &a, const float3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator/ (const float3 &a, const float s)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 operator/ (const float s, const float3 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator/= (float3 &a, const float s)`

- `OPTIXU_INLINE RT_HOSTDEVICE float3 clamp (const float3 &v, const float a, const float b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 clamp (const float3 &v, const float3 &a, const float3 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE float4 make_float4 (const float s)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 make_float4 (const float3 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 make_float4 (const int4 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 make_float4 (const uint4 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE float4 fminf (const float4 &a, const float4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float fminf (const float4 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE float4 fmaxf (const float4 &a, const float4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float fmaxf (const float4 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE float4 operator+ (const float4 &a, const float4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 operator+ (const float4 &a, const float b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 operator+ (const float a, const float4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator+= (float4 &a, const float4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE float4 operator- (const float4 &a, const float4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 operator- (const float4 &a, const float b)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 operator- (const float a, const float4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator-= (float4 &a, const float4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE float4 operator* (const float4 &a, const float4 &s)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 operator* (const float4 &a, const float s)`
- `OPTIXU_INLINE RT_HOSTDEVICE float4 operator* (const float s, const float4 &a)`

- OPTIXU_INLINE RT_HOSTDEVICE void operator*=(float4 &a, const float4 &s)
- OPTIXU_INLINE RT_HOSTDEVICE void operator*=(float4 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float4 operator/ (const float4 &a, const float4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE float4 operator/ (const float4 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float4 operator/ (const float s, const float4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void operator/= (float4 &a, const float s)
- OPTIXU_INLINE RT_HOSTDEVICE float4 clamp (const float4 &v, const float a, const float b)
- OPTIXU_INLINE RT_HOSTDEVICE float4 clamp (const float4 &v, const float4 &a, const float4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int2 make_int2 (const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int2 make_int2 (const float2 &a)
- OPTIXU_INLINE RT_HOSTDEVICE int2 operator+ (const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void operator+= (int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int2 operator- (const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int2 operator- (const int2 &a, const int b)
- OPTIXU_INLINE RT_HOSTDEVICE void operator-= (int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int2 operator* (const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int2 operator* (const int2 &a, const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int2 operator* (const int s, const int2 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void operator*= (int2 &a, const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int2 clamp (const int2 &v, const int a, const int b)
- OPTIXU_INLINE RT_HOSTDEVICE int2 clamp (const int2 &v, const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE bool operator==(const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE bool operator!=(const int2 &a, const int2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int3 make_int3 (const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int3 make_int3 (const float3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE int3 operator+ (const int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void operator+= (int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int3 operator- (const int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void operator-= (int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int3 operator* (const int3 &a, const int3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE int3 operator* (const int3 &a, const int s)
- OPTIXU_INLINE RT_HOSTDEVICE int3 operator* (const int s, const int3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void operator*= (int3 &a, const int s)

- `OPTIXU_INLINE RT_HOSTDEVICE int3 operator/ (const int3 &a, const int3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int3 operator/ (const int3 &a, const int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE int3 operator/ (const int s, const int3 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator/= (int3 &a, const int s)`

- `OPTIXU_INLINE RT_HOSTDEVICE int3 clamp (const int3 &v, const int a, const int b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int3 clamp (const int3 &v, const int3 &a, const int3 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE bool operator== (const int3 &a, const int3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE bool operator!= (const int3 &a, const int3 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE int4 make_int4 (const int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 make_int4 (const float4 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE int4 operator+ (const int4 &a, const int4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator+= (int4 &a, const int4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE int4 operator- (const int4 &a, const int4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator-= (int4 &a, const int4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE int4 operator* (const int4 &a, const int4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 operator* (const int4 &a, const int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 operator* (const int s, const int4 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator*= (int4 &a, const int s)`

- `OPTIXU_INLINE RT_HOSTDEVICE int4 operator/ (const int4 &a, const int4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 operator/ (const int4 &a, const int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 operator/ (const int s, const int4 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator/= (int4 &a, const int s)`

- `OPTIXU_INLINE RT_HOSTDEVICE int4 clamp (const int4 &v, const int a, const int b)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 clamp (const int4 &v, const int4 &a, const int4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE bool operator== (const int4 &a, const int4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE bool operator!= (const int4 &a, const int4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint2 make_uint2 (const unsigned int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint2 make_uint2 (const float2 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint2 operator+ (const uint2 &a, const uint2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator+= (uint2 &a, const uint2 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint2 operator- (const uint2 &a, const uint2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint2 operator- (const uint2 &a, const unsigned int b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator-= (uint2 &a, const uint2 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint2 operator* (const uint2 &a, const uint2 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint2 operator* (const uint2 &a, const unsigned int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint2 operator* (const unsigned int s, const uint2 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator*= (uint2 &a, const unsigned int s)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint2 clamp (const uint2 &v, const unsigned int a, const unsigned int b)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint2 clamp (const uint2 &v, const uint2 &a, const uint2 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE bool operator== (const uint2 &a, const uint2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE bool operator!= (const uint2 &a, const uint2 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint3 make_uint3 (const unsigned int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint3 make_uint3 (const float3 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint3 operator+ (const uint3 &a, const uint3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator+= (uint3 &a, const uint3 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint3 operator- (const uint3 &a, const uint3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator-= (uint3 &a, const uint3 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint3 operator* (const uint3 &a, const uint3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint3 operator* (const uint3 &a, const unsigned int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint3 operator* (const unsigned int s, const uint3 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator*= (uint3 &a, const unsigned int s)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint3 operator/ (const uint3 &a, const uint3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint3 operator/ (const uint3 &a, const unsigned int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint3 operator/ (const unsigned int s, const uint3 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator/= (uint3 &a, const unsigned int s)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint3 clamp (const uint3 &v, const unsigned int a, const unsigned int b)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint3 clamp (const uint3 &v, const uint3 &a, const uint3 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE bool operator== (const uint3 &a, const uint3 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE bool operator!= (const uint3 &a, const uint3 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint4 make_uint4 (const unsigned int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint4 make_uint4 (const float4 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint4 min (const uint4 &a, const uint4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint4 max (const uint4 &a, const uint4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint4 operator+ (const uint4 &a, const uint4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator+= (uint4 &a, const uint4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint4 operator- (const uint4 &a, const uint4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator-= (uint4 &a, const uint4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint4 operator* (const uint4 &a, const uint4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint4 operator* (const uint4 &a, const unsigned int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint4 operator* (const unsigned int s, const uint4 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator*= (uint4 &a, const unsigned int s)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint4 operator/ (const uint4 &a, const uint4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint4 operator/ (const uint4 &a, const unsigned int s)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint4 operator/ (const unsigned int s, const uint4 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator/= (uint4 &a, const unsigned int s)`

- `OPTIXU_INLINE RT_HOSTDEVICE uint4 clamp (const uint4 &v, const unsigned int a, const unsigned int b)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint4 clamp (const uint4 &v, const uint4 &a, const uint4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE bool operator== (const uint4 &a, const uint4 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE bool operator!= (const uint4 &a, const uint4 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 make_longlong2 (const long long s)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 make_longlong2 (const float2 &a)`

- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 operator+ (const longlong2 &a, const longlong2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator+= (longlong2 &a, const longlong2 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 operator- (const longlong2 &a, const longlong2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 operator- (const longlong2 &a, const long long b)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator-= (longlong2 &a, const longlong2 &b)`

- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 operator* (const longlong2 &a, const longlong2 &b)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 operator* (const longlong2 &a, const long long s)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 operator* (const long long s, const longlong2 &a)`
- `OPTIXU_INLINE RT_HOSTDEVICE void operator*= (longlong2 &a, const long long s)`

- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 clamp (const longlong2 &v, const long long a, const long long b)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 clamp (const longlong2 &v, const longlong2 &a, const longlong2 &b)`

- OPTIXU_INLINE RT_HOSTDEVICE bool operator==(const longlong2 &a, const longlong2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE bool operator!=(const longlong2 &a, const longlong2 &b)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 make_longlong3(const long long s)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 make_longlong3(const float3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 operator+(const longlong3 &a, const longlong3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void operator+=(longlong3 &a, const longlong3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 operator-(const longlong3 &a, const longlong3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void operator-=(longlong3 &a, const longlong3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 operator*(const longlong3 &a, const longlong3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 operator*(const longlong3 &a, const long long s)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 operator*(const long long s, const longlong3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void operator*=(longlong3 &a, const long long s)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 operator/(const longlong3 &a, const longlong3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 operator/(const longlong3 &a, const long long s)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 operator/(const long long s, const longlong3 &a)
- OPTIXU_INLINE RT_HOSTDEVICE void operator/=(longlong3 &a, const long long s)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 clamp(const longlong3 &v, const long long a, const long long b)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong3 clamp(const longlong3 &v, const longlong3 &a, const longlong3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE bool operator==(const longlong3 &a, const longlong3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE bool operator!=(const longlong3 &a, const longlong3 &b)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong4 make_longlong4(const long long s)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong4 make_longlong4(const float4 &a)
- OPTIXU_INLINE RT_HOSTDEVICE
longlong4 operator+(const longlong4 &a, const longlong4 &b)
- OPTIXU_INLINE RT_HOSTDEVICE void operator+=(longlong4 &a, const longlong4 &b)

- **OPTIXU_INLINE RT_HOSTDEVICE**
longlong4 **operator-** (const longlong4 &a, const longlong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator-=** (longlong4 &a, const longlong4 &b)

- **OPTIXU_INLINE RT_HOSTDEVICE**
longlong4 **operator*** (const longlong4 &a, const longlong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
longlong4 **operator*** (const longlong4 &a, const long long s)
- **OPTIXU_INLINE RT_HOSTDEVICE**
longlong4 **operator*** (const long long s, const longlong4 &a)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator*=** (longlong4 &a, const long long s)

- **OPTIXU_INLINE RT_HOSTDEVICE**
longlong4 **operator/** (const longlong4 &a, const longlong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
longlong4 **operator/** (const longlong4 &a, const long long s)
- **OPTIXU_INLINE RT_HOSTDEVICE**
longlong4 **operator/** (const long long s, const longlong4 &a)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator/=** (longlong4 &a, const long long s)

- **OPTIXU_INLINE RT_HOSTDEVICE**
longlong4 **clamp** (const longlong4 &v, const long long a, const long long b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
longlong4 **clamp** (const longlong4 &v, const longlong4 &a, const longlong4 &b)

- **OPTIXU_INLINE RT_HOSTDEVICE bool operator==** (const longlong4 &a, const longlong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE bool operator!=** (const longlong4 &a, const longlong4 &b)

- **OPTIXU_INLINE RT_HOSTDEVICE**
ulonglong2 **make_ulonglong2** (const unsigned long long s)
- **OPTIXU_INLINE RT_HOSTDEVICE**
ulonglong2 **make_ulonglong2** (const float2 &a)

- **OPTIXU_INLINE RT_HOSTDEVICE**
ulonglong2 **operator+** (const ulonglong2 &a, const ulonglong2 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator+=** (ulonglong2 &a, const ulonglong2 &b)

- **OPTIXU_INLINE RT_HOSTDEVICE**
ulonglong2 **operator-** (const ulonglong2 &a, const ulonglong2 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
ulonglong2 **operator-** (const ulonglong2 &a, const unsigned long long b)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator-=** (ulonglong2 &a, const ulonglong2 &b)

- **OPTIXU_INLINE RT_HOSTDEVICE**
ulonglong2 **operator*** (const ulonglong2 &a, const ulonglong2 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
ulonglong2 **operator*** (const ulonglong2 &a, const unsigned long long s)

- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong2 **operator*** (const unsigned long long s, const ulonglong2 &a)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator*=(ulonglong2 &a, const unsigned long long s)**

- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong2 **clamp** (const ulonglong2 &v, const unsigned long long a, const unsigned long long b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong2 **clamp** (const ulonglong2 &v, const ulonglong2 &a, const ulonglong2 &b)

- **OPTIXU_INLINE RT_HOSTDEVICE bool operator==(const ulonglong2 &a, const ulonglong2 &b)**
- **OPTIXU_INLINE RT_HOSTDEVICE bool operator!=(const ulonglong2 &a, const ulonglong2 &b)**

- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **make_ulonglong3** (const unsigned long long s)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **make_ulonglong3** (const float3 &a)

- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **operator+** (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator+=(ulonglong3 &a, const ulonglong3 &b)**

- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **operator-** (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator-=(ulonglong3 &a, const ulonglong3 &b)**

- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **operator*** (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **operator*** (const ulonglong3 &a, const unsigned long long s)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **operator*** (const unsigned long long s, const ulonglong3 &a)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator*=(ulonglong3 &a, const unsigned long long s)**

- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **operator/** (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **operator/** (const ulonglong3 &a, const unsigned long long s)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **operator/** (const unsigned long long s, const ulonglong3 &a)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator/=(ulonglong3 &a, const unsigned long long s)**

- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **clamp** (const ulonglong3 &v, const unsigned long long a, const unsigned long long b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong3 **clamp** (const ulonglong3 &v, const ulonglong3 &a, const ulonglong3 &b)

- **OPTIXU_INLINE RT_HOSTDEVICE bool operator==(const ulonglong3 &a, const ulonglong3 &b)**
- **OPTIXU_INLINE RT_HOSTDEVICE bool operator!=(const ulonglong3 &a, const ulonglong3 &b)**

- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **make_ulonglong4** (const unsigned long long s)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **make_ulonglong4** (const float4 &a)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **min** (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **max** (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **operator+** (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator+=** (ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **operator-** (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator-=** (ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **operator*** (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **operator*** (const ulonglong4 &a, const unsigned long long s)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **operator*** (const unsigned long long s, const ulonglong4 &a)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator*=** (ulonglong4 &a, const unsigned long long s)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **operator/** (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **operator/** (const ulonglong4 &a, const unsigned long long s)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **operator/** (const unsigned long long s, const ulonglong4 &a)
- **OPTIXU_INLINE RT_HOSTDEVICE void operator/=** (ulonglong4 &a, const unsigned long long s)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **clamp** (const ulonglong4 &v, const unsigned long long a, const unsigned long long b)
- **OPTIXU_INLINE RT_HOSTDEVICE**
 - ulonglong4 **clamp** (const ulonglong4 &v, const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE bool operator==** (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE bool operator!=** (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU_INLINE RT_HOSTDEVICE int2 make_int2** (const int3 &v0)
- **OPTIXU_INLINE RT_HOSTDEVICE int2 make_int2** (const int4 &v0)
- **OPTIXU_INLINE RT_HOSTDEVICE int3 make_int3** (const int4 &v0)
- **OPTIXU_INLINE RT_HOSTDEVICE uint2 make_uint2** (const uint3 &v0)

- `OPTIXU_INLINE RT_HOSTDEVICE uint2 make_uint2 (const uint4 &v0)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint3 make_uint3 (const uint4 &v0)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 make_longlong2 (const longlong3 &v0)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong2 make_longlong2 (const longlong4 &v0)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong3 make_longlong3 (const longlong4 &v0)`
- `OPTIXU_INLINE RT_HOSTDEVICE
ulonglong2 make_ulonglong2 (const ulonglong3 &v0)`
- `OPTIXU_INLINE RT_HOSTDEVICE
ulonglong2 make_ulonglong2 (const ulonglong4 &v0)`
- `OPTIXU_INLINE RT_HOSTDEVICE
ulonglong3 make_ulonglong3 (const ulonglong4 &v0)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 make_float2 (const float3 &v0)`
- `OPTIXU_INLINE RT_HOSTDEVICE float2 make_float2 (const float4 &v0)`
- `OPTIXU_INLINE RT_HOSTDEVICE float3 make_float3 (const float4 &v0)`

- `OPTIXU_INLINE RT_HOSTDEVICE int3 make_int3 (const int v0, const int2 &v1)`
- `OPTIXU_INLINE RT_HOSTDEVICE int3 make_int3 (const int2 &v0, const int v1)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 make_int4 (const int v0, const int v1, const int2 &v2)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 make_int4 (const int v0, const int2 &v1, const int v2)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 make_int4 (const int2 &v0, const int v1, const int v2)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 make_int4 (const int v0, const int3 &v1)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 make_int4 (const int3 &v0, const int v1)`
- `OPTIXU_INLINE RT_HOSTDEVICE int4 make_int4 (const int2 &v0, const int2 &v1)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint3 make_uint3 (const unsigned int v0, const uint2 &v1)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint3 make_uint3 (const uint2 &v0, const unsigned int v1)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint4 make_uint4 (const unsigned int v0, const unsigned int v1, const uint2 &v2)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint4 make_uint4 (const unsigned int v0, const uint2 &v1, const unsigned int v2)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint4 make_uint4 (const uint2 &v0, const unsigned int v1, const unsigned int v2)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint4 make_uint4 (const uint3 &v0, const unsigned int v1, const unsigned int v2)`
- `OPTIXU_INLINE RT_HOSTDEVICE uint4 make_uint4 (const uint2 &v0, const uint2 &v1, const unsigned int v2)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong3 make_longlong3 (const long long v0, const longlong2 &v1)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong3 make_longlong3 (const longlong2 &v0, const long long v1)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong4 make_longlong4 (const long long v0, const long long v1, const longlong2 &v2)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong4 make_longlong4 (const long long v0, const longlong2 &v1, const long long v2)`
- `OPTIXU_INLINE RT_HOSTDEVICE
longlong4 make_longlong4 (const longlong2 &v0, const long long v1, const long long v2)`

- **OPTIXU_INLINE RT_HOSTDEVICE**
`longlong4 make_longlong4 (const long long v0, const longlong3 &v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`longlong4 make_longlong4 (const longlong3 &v0, const long long v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`longlong4 make_longlong4 (const longlong2 &v0, const longlong2 &v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`ulonglong3 make_ulonglong3 (const unsigned long long v0, const ulonglong2 &v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`ulonglong3 make_ulonglong3 (const ulonglong2 &v0, const unsigned long long v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`ulonglong4 make_ulonglong4 (const unsigned long long v0, const unsigned long long v1, const ulonglong2 &v2)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`ulonglong4 make_ulonglong4 (const unsigned long long v0, const ulonglong2 &v1, const unsigned long long v2)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`ulonglong4 make_ulonglong4 (const ulonglong2 &v0, const unsigned long long v1, const unsigned long long v2)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`ulonglong4 make_ulonglong4 (const unsigned long long v0, const ulonglong3 &v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`ulonglong4 make_ulonglong4 (const ulonglong3 &v0, const unsigned long long v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`ulonglong4 make_ulonglong4 (const ulonglong2 &v0, const ulonglong2 &v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`float3 make_float3 (const float2 &v0, const float v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`float3 make_float3 (const float v0, const float2 &v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`float4 make_float4 (const float v0, const float v1, const float2 &v2)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`float4 make_float4 (const float v0, const float2 &v1, const float v2)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`float4 make_float4 (const float2 &v0, const float v1, const float v2)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`float4 make_float4 (const float v0, const float3 &v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`float4 make_float4 (const float3 &v0, const float v1)`
- **OPTIXU_INLINE RT_HOSTDEVICE**
`float4 make_float4 (const float2 &v0, const float2 &v1)`
- **std::ostream & operator<<** (**std::ostream** &os, const **optix::float4** &v)
• **std::istream & operator>>** (**std::istream** &is, **optix::float4** &v)
• **std::ostream & operator<<** (**std::ostream** &os, const **optix::float3** &v)
• **std::istream & operator>>** (**std::istream** &is, **optix::float3** &v)
• **std::ostream & operator<<** (**std::ostream** &os, const **optix::float2** &v)
• **std::istream & operator>>** (**std::istream** &is, **optix::float2** &v)

• **std::ostream & operator<<** (**std::ostream** &os, const **optix::int4** &v)
• **std::istream & operator>>** (**std::istream** &is, **optix::int4** &v)
• **std::ostream & operator<<** (**std::ostream** &os, const **optix::int3** &v)

- **std::istream & operator>>** (**std::istream** &is, optix::int3 &v)
- **std::ostream & operator<<** (**std::ostream** &os, const optix::int2 &v)
- **std::istream & operator>>** (**std::istream** &is, optix::int2 &v)

- **std::ostream & operator<<** (**std::ostream** &os, const optix::uint4 &v)
- **std::istream & operator>>** (**std::istream** &is, optix::uint4 &v)
- **std::ostream & operator<<** (**std::ostream** &os, const optix::uint3 &v)
- **std::istream & operator>>** (**std::istream** &is, optix::uint3 &v)
- **std::ostream & operator<<** (**std::ostream** &os, const optix::uint2 &v)
- **std::istream & operator>>** (**std::istream** &is, optix::uint2 &v)

- template<unsigned int M, unsigned int N>
std::ostream & operator<< (**std::ostream** &os, const optix::Matrix< M, N > &m)
- template<unsigned int M, unsigned int N>
std::istream & operator>> (**std::istream** &is, optix::Matrix< M, N > &m)

Variables

- const unsigned char **KEY** [7] = {164, 195, 147, 255, 203, 161, 184}

- **rtTextureId id**
- **rtTextureId float x**
- * **retVal** = tmp
- **rtTextureId float float y**
- **rtTextureId float float float z**
- **rtTextureId float float int comp**
- **rtTextureId float float dPdx**
- **rtTextureId float float float dPdy**
- **rtTextureId float int layer**
- **rtTextureId float float level**
- **__device__ uint3 rtTexSize (rtTextureId id)**
- template<typename T >
__device__ T rtTex1D (rtTextureId id, float x)
- template<>
__device__ float4 rtTex1D (rtTextureId id, float x)
- template<>
__device__ int4 rtTex1D (rtTextureId id, float x)
- template<>
__device__ uint4 rtTex1D (rtTextureId id, float x)
- **_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 rtTex1DFetch (rtTextureId id, int x)
- template<>
__device__ float4 rtTex1DFetch (rtTextureId id, int x)
- template<>
__device__ int4 rtTex1DFetch (rtTextureId id, int x)

- template<>
 __device__ uint4 rtTex1DFetch (rtTextureId id, int x)
- _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 rtTex2D (rtTextureId id, float x, float y)
- template<>
 __device__ float4 rtTex2D (rtTextureId id, float x, float y)
- template<>
 __device__ int4 rtTex2D (rtTextureId id, float x, float y)
- template<>
 __device__ uint4 rtTex2D (rtTextureId id, float x, float y)
- _OPTIX_TEX_FUNC_DECLARE_(rtTex2D,(rtTextureId id, float x, float y),(id, x, y)) template< typename T > inline __device__ void rtTex2D(T *retVal)
- template<typename T >
 __device__ T rtTex2DFetch (rtTextureId id, int x, int y)
- template<>
 __device__ float4 rtTex2DFetch (rtTextureId id, int x, int y)
- template<>
 __device__ int4 rtTex2DFetch (rtTextureId id, int x, int y)
- template<>
 __device__ uint4 rtTex2DFetch (rtTextureId id, int x, int y)
- _OPTIX_TEX_FUNC_DECLARE_(rtTex2DFetch,(rtTextureId id, int x, int y),(id, x, y)) template< typename T > inline __device__ void rtTex2DFetch(T *retVal)
- template<typename T >
 __device__ T rtTex3D (rtTextureId id, float x, float y, float z)
- template<>
 __device__ float4 rtTex3D (rtTextureId id, float x, float y, float z)
- template<>
 __device__ int4 rtTex3D (rtTextureId id, float x, float y, float z)
- template<>
 __device__ uint4 rtTex3D (rtTextureId id, float x, float y, float z)
- _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 rtTex3DFetch (rtTextureId id, int x, int y, int z)
- template<>
 __device__ float4 rtTex3DFetch (rtTextureId id, int x, int y, int z)
- template<>
 __device__ int4 rtTex3DFetch (rtTextureId id, int x, int y, int z)
- template<>
 __device__ uint4 rtTex3DFetch (rtTextureId id, int x, int y, int z)
- _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 rtTex2DGather (rtTextureId id, float x, float y, int comp=0)
- template<>
 __device__ float4 rtTex2DGather (rtTextureId id, float x, float y, int comp)

- template<>
 __device__ int4 rtTex2DGather (rtTextureId id, float x, float y, int comp)
- template<>
 __device__ uint4 rtTex2DGather (rtTextureId id, float x, float y, int comp)
- _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 rtTex1DGrad (rtTextureId id, float x, float dPdx, float dPdy)
- template<>
 __device__ int4 rtTex1DGrad (rtTextureId id, float x, float dPdx, float dPdy)
- template<>
 __device__ uint4 rtTex1DGrad (rtTextureId id, float x, float dPdx, float dPdy)
- _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 rtTex2DGrad (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)
- template<>
 __device__ float4 rtTex2DGrad (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)
- template<>
 __device__ int4 rtTex2DGrad (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)
- template<>
 __device__ uint4 rtTex2DGrad (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy)
- _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 rtTex3DGrad (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)
- template<>
 __device__ float4 rtTex3DGrad (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)
- template<>
 __device__ int4 rtTex3DGrad (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)
- template<>
 __device__ uint4 rtTex3DGrad (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy)
- _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 rtTex1DLayeredGrad (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- template<>
 __device__ float4 rtTex1DLayeredGrad (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- template<>
 __device__ int4 rtTex1DLayeredGrad (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- template<>
 __device__ uint4 rtTex1DLayeredGrad (rtTextureId id, float x, int layer, float dPdx, float dPdy)
- _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 rtTex2DLayeredGrad (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- template<>
 __device__ float4 rtTex2DLayeredGrad (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- template<>
 __device__ int4 rtTex2DLayeredGrad (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- template<>
 __device__ uint4 rtTex2DLayeredGrad (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy)
- _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 rtTex1DLod (rtTextureId id, float x, float level)
- template<>
 __device__ float4 rtTex1DLod (rtTextureId id, float x, float level)
- template<>
 __device__ int4 rtTex1DLod (rtTextureId id, float x, float level)
- template<>
 __device__ uint4 rtTex1DLod (rtTextureId id, float x, float level)
- _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 rtTex2DLod (rtTextureId id, float x, float y, float level)
- template<>
 __device__ float4 rtTex2DLod (rtTextureId id, float x, float y, float level)
- template<>
 __device__ int4 rtTex2DLod (rtTextureId id, float x, float y, float level)
- template<>
 __device__ uint4 rtTex2DLod (rtTextureId id, float x, float y, float level)
- _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 rtTex3DLod (rtTextureId id, float x, float y, float z, float level)
- template<>
 __device__ float4 rtTex3DLod (rtTextureId id, float x, float y, float z, float level)
- template<>
 __device__ int4 rtTex3DLod (rtTextureId id, float x, float y, float z, float level)
- template<>
 __device__ uint4 rtTex3DLod (rtTextureId id, float x, float y, float z, float level)
- _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 rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)

- template<>
 `__device__ float4 rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)`
- template<>
 `__device__ int4 rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)`
- template<>
 `__device__ uint4 rtTex1DLayeredLod (rtTextureId id, float x, int layer, float level)`
- `_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 rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- template<>
 `__device__ float4 rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- template<>
 `__device__ int4 rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- template<>
 `__device__ uint4 rtTex2DLayeredLod (rtTextureId id, float x, float y, int layer, float level)`
- `_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 rtTex1DLayered (rtTextureId id, float x, int layer)`
- template<>
 `__device__ float4 rtTex1DLayered (rtTextureId id, float x, int layer)`
- template<>
 `__device__ int4 rtTex1DLayered (rtTextureId id, float x, int layer)`
- template<>
 `__device__ uint4 rtTex1DLayered (rtTextureId id, float x, int layer)`
- `_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 rtTex2DLayered (rtTextureId id, float x, float y, int layer)`
- template<>
 `__device__ float4 rtTex2DLayered (rtTextureId id, float x, float y, int layer)`
- template<>
 `__device__ int4 rtTex2DLayered (rtTextureId id, float x, float y, int layer)`
- template<>
 `__device__ uint4 rtTex2DLayered (rtTextureId id, float x, float y, int layer)`
- `_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 rtTexCubemap (rtTextureId id, float x, float y, float z)`
- template<>
 `__device__ float4 rtTexCubemap (rtTextureId id, float x, float y, float z)`
- template<>
 `__device__ int4 rtTexCubemap (rtTextureId id, float x, float y, float z)`
- template<>
 `__device__ uint4 rtTexCubemap (rtTextureId id, float x, float y, float z)`

- `_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 rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`
- `template<>`
`__device__ float4 rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`
- `template<>`
`__device__ int4 rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`
- `template<>`
`__device__ uint4 rtTexCubemapLayered (rtTextureId id, float x, float y, float z, int layer)`
- `_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 rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`
`__device__ float4 rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`
`__device__ int4 rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- `template<>`
`__device__ uint4 rtTexCubemapLod (rtTextureId id, float x, float y, float z, float level)`
- `_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 rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- `template<>`
`__device__ float4 rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- `template<>`
`__device__ int4 rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- `template<>`
`__device__ uint4 rtTexCubemapLayeredLod (rtTextureId id, float x, float y, float z, int layer, float level)`
- `_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`

6.1.1 Typedef Documentation

- 6.1.1.1 **typedef Matrix<2, 2> optix::Matrix2x2**
- 6.1.1.2 **typedef Matrix<2, 3> optix::Matrix2x3**
- 6.1.1.3 **typedef Matrix<2, 4> optix::Matrix2x4**
- 6.1.1.4 **typedef Matrix<3, 2> optix::Matrix3x2**
- 6.1.1.5 **typedef Matrix<3, 3> optix::Matrix3x3**
- 6.1.1.6 **typedef Matrix<3, 4> optix::Matrix3x4**
- 6.1.1.7 **typedef Matrix<4, 2> optix::Matrix4x2**
- 6.1.1.8 **typedef Matrix<4, 3> optix::Matrix4x3**
- 6.1.1.9 **typedef Matrix<4, 4> optix::Matrix4x4**
- 6.1.1.10 **typedef size_t optix::optix_size_t**
- 6.1.1.11 **typedef int optix::rtTextureId**
- 6.1.1.12 **typedef unsigned int optix::uint**
- 6.1.1.13 **typedef unsigned short optix::ushort**

6.1.2 Enumeration Type Documentation

6.1.2.1 enum optix::ObjectStorageType

Enumerator

OBJECT_STORAGE_CONSTANT
OBJECT_STORAGE_SHARED
OBJECT_STORAGE_GLOBAL
OBJECT_STORAGE_LINEAR_TEXTURE
OBJECT_STORAGE_BLOCKED_TEXTURE

6.1.2.2 enum optix::rtiTexLookupKind

Enumerator

TEX_LOOKUP_1D
TEX_LOOKUP_2D
TEX_LOOKUP_3D
TEX_LOOKUP_A1
TEX_LOOKUP_A2

TEX_LOOKUP_CUBE***TEX_LOOKUP_ACUBE***

6.1.3 Function Documentation

**6.1.3.1 `optix::_OPTIX_TEX_FUNC_DECLARE_(
 rtTex1D,
 (rtTextureId id, float x),
 (id, x))`**

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

**6.1.3.2 `optix::_OPTIX_TEX_FUNC_DECLARE_(
 rtTex1DFetch,
 (rtTextureId id, int x),`**

```
(id, x) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.3 optix::_OPTIX_TEX_FUNC_DECLARE_()
    rtTex2D ,
    (rtTextureId id, float x, float y) ,
    (id, x, y) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size

reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.4 optix::_OPTIX_TEX_FUNC_DECLARE_(

```
rtTex2DFetch ,
(rtTextureId id, int x, int y) ,
(id, x, y) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and

rtTexCubemapLayeredLod.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.5 optix::OPTIX_TEX_FUNC_DECLARE_(
    rtTex3D ,
    (rtTextureId id, float x, float y, float z) ,
    (id, x, y, z) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**,

rtTex2DLod, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.6 optix::_OPTIX_TEX_FUNC_DECLARE_(
rtTex3DFetch ,
(rtTextureId id, int x, int y, int z) ,
(id, x, y, z))

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.7 optix::_OPTIX_TEX_FUNC_DECLARE_(
rtTex2DGather ,
(rtTextureId id, float x, float y, int comp) ,

(id, x, y, comp))

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.8 optix::_OPTIX_TEX_FUNC_DECLARE_(

```
    rtTex1DGrad ,
    (rtTextureId id, float x, float dPdx, float dPdy) ,
    (id, x, dPdx, dPdy) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size

reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.9 optix::_OPTIX_TEX_FUNC_DECLARE_(

```
    rtTex2DGrad ,
    (rtTextureId id, float x, float y, float2 dPdx, float2 dPdy) ,
    (id, x, y, dPdx, dPdy) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and

rtTexCubemapLayeredLod.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.10 optix::_OPTIX_TEX_FUNC_DECLARE_(

```
    rtTex3DGrad ,
    (rtTextureId id, float x, float y, float z, float4 dPdx, float4 dPdy) ,
    (id, x, y, z, dPdx, dPdy) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**,

rtTex2DLod, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.11 optix::_OPTIX_TEX_FUNC_DECLARE_ (
    rtTex1DLayeredGrad ,
    (rtTextureId id, float x, int layer, float dPdx, float dPdy) ,
    (id, x, layer, dPdx, dPdy) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.12 optix::_OPTIX_TEX_FUNC_DECLARE_ (
    rtTex2DLayeredGrad ,
    (rtTextureId id, float x, float y, int layer, float2 dPdx, float2 dPdy) ,
```

```
(id, x, y, layer, dPdx, dPdy) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.13 optix::_OPTIX_TEX_FUNC_DECLARE_(
    rtTex1DLod ,
    (rtTextureId id, float x, float level) ,
    (id, x, level) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size

reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.14 optix::_OPTIX_TEX_FUNC_DECLARE_(

```
    rtTex2DLod ,
    (rtTextureId id, float x, float y, float level) ,
    (id, x, y, level) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and

rtTexCubemapLayeredLod.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.15 optix::_OPTIX_TEX_FUNC_DECLARE_(

```
    rtTex3DLod ,
    (rtTextureId id, float x, float y, float z, float level) ,
    (id, x, y, z, level) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**,

rtTex2DLod, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.16 optix::_OPTIX_TEX_FUNC_DECLARE_ (
    rtTex1DLayeredLod ,
    (rtTextureId id, float x, int layer, float level) ,
    (id, x, layer, level) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.17 optix::_OPTIX_TEX_FUNC_DECLARE_ (
    rtTex2DLayeredLod ,
    (rtTextureId id, float x, float y, int layer, float level) ,
```

(id, x, y, layer, level))

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.18 optix::_OPTIX_TEX_FUNC_DECLARE_ (

```
    rtTex1DLayered ,
    (rtTextureId id, float x, int layer) ,
    (id, x, layer) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size

reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.19 optix::_OPTIX_TEX_FUNC_DECLARE_ (

```
    rtTex2DLayered ,
    (rtTextureId id, float x, float y, int layer) ,
    (id, x, y, layer) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and

rtTexCubemapLayeredLod.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.20 optix::_OPTIX_TEX_FUNC_DECLARE_(
    rtTexCubemap ,
    (rtTextureId id, float x, float y, float z) ,
    (id, x, y, z) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**,

rtTex2DLod, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.21 optix::_OPTIX_TEX_FUNC_DECLARE_ (
    rtTexCubemapLayered ,
    (rtTextureId id, float x, float y, float z, int layer) ,
    (id, x, y, z, layer) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.22 optix::_OPTIX_TEX_FUNC_DECLARE_ (
    rtTexCubemapLod ,
    (rtTextureId id, float x, float y, float z, float level) ,
```

(id, x, y, z, level))

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.23 optix::_OPTIX_TEX_FUNC_DECLARE_()
    rtTexCubemapLayeredLod ,
    (rtTextureId id, float x, float y, float z, int layer, float level) ,
    (id, x, y, z, layer, level) )
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size

reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.24 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 )
```

bilerp

6.1.3.25 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 )
```

bilerp

6.1.3.26 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 )
```

bilerp

6.1.3.27 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 )
```

bilerp

6.1.3.28 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::cart_to_pol (

```
    const float3 & v )
```

Convert Cartesian coordinates to polar coordinates.

6.1.3.29 OPTIXU_INLINE RT_HOSTDEVICE float optix::clamp (

```
    const float f,
    const float a,
    const float b )
```

clamp

6.1.3.30 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::clamp (

```
    const float2 & v,
    const float a,
    const float b )
```

clamp

6.1.3.31 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::clamp (

```
    const float2 & v,
    const float2 & a,
    const float2 & b )
```

clamp

6.1.3.32 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::clamp (

```
    const float3 & v,
    const float a,
```

const float *b*)

clamp

6.1.3.33 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::clamp (
const float3 & *v*,
const float3 & *a*,
const float3 & *b*)

clamp

6.1.3.34 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::clamp (
const float4 & *v*,
const float *a*,
const float *b*)

clamp

6.1.3.35 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::clamp (
const float4 & *v*,
const float4 & *a*,
const float4 & *b*)

clamp

6.1.3.36 OPTIXU_INLINE RT_HOSTDEVICE int optix::clamp (
const int *f*,
const int *a*,
const int *b*)

clamp

6.1.3.37 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::clamp (
const int2 & *v*,
const int *a*,
const int *b*)

clamp

6.1.3.38 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::clamp (
const int2 & *v*,
const int2 & *a*,
const int2 & *b*)

clamp

6.1.3.39 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::clamp (
const int3 & *v*,

```
    const int a,  
    const int b )
```

clamp

6.1.3.40 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::clamp (

```
    const int3 & v,  
    const int3 & a,  
    const int3 & b )
```

clamp

6.1.3.41 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::clamp (

```
    const int4 & v,  
    const int a,  
    const int b )
```

clamp

6.1.3.42 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::clamp (

```
    const int4 & v,  
    const int4 & a,  
    const int4 & b )
```

clamp

6.1.3.43 OPTIXU_INLINE RT_HOSTDEVICE unsigned int optix::clamp (

```
    const unsigned int f,  
    const unsigned int a,  
    const unsigned int b )
```

clamp

6.1.3.44 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::clamp (

```
    const uint2 & v,  
    const unsigned int a,  
    const unsigned int b )
```

clamp

6.1.3.45 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::clamp (

```
    const uint2 & v,  
    const uint2 & a,  
    const uint2 & b )
```

clamp

6.1.3.46 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::clamp (

```
    const uint3 & v,  
    const unsigned int a,  
    const unsigned int b )
```

clamp

6.1.3.47 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::clamp (

```
    const uint3 & v,  
    const uint3 & a,  
    const uint3 & b )
```

clamp

6.1.3.48 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::clamp (

```
    const uint4 & v,  
    const unsigned int a,  
    const unsigned int b )
```

clamp

6.1.3.49 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::clamp (

```
    const uint4 & v,  
    const uint4 & a,  
    const uint4 & b )
```

clamp

6.1.3.50 OPTIXU_INLINE RT_HOSTDEVICE long long optix::clamp (

```
    const long long f,  
    const long long a,  
    const long long b )
```

clamp

6.1.3.51 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::clamp (

```
    const longlong2 & v,  
    const long long a,  
    const long long b )
```

clamp

6.1.3.52 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::clamp (

```
    const longlong2 & v,  
    const longlong2 & a,  
    const longlong2 & b )
```

clamp

6.1.3.53 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::clamp (
 const longlong3 & v,
 const long long a,
 const long long b)

clamp

6.1.3.54 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::clamp (
 const longlong3 & v,
 const longlong3 & a,
 const longlong3 & b)

clamp

6.1.3.55 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::clamp (
 const longlong4 & v,
 const long long a,
 const long long b)

clamp

6.1.3.56 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::clamp (
 const longlong4 & v,
 const longlong4 & a,
 const longlong4 & b)

clamp

6.1.3.57 OPTIXU_INLINE RT_HOSTDEVICE unsigned long long optix::clamp (
 const unsigned long long f,
 const unsigned long long a,
 const unsigned long long b)

clamp

6.1.3.58 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::clamp (
 const ulonglong2 & v,
 const unsigned long long a,
 const unsigned long long b)

clamp

6.1.3.59 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::clamp (
 const ulonglong2 & v,
 const ulonglong2 & a,

const ulonglong2 & b)

clamp

6.1.3.60 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::clamp (

const ulonglong3 & v,
const unsigned long long a,
const unsigned long long b)

clamp

6.1.3.61 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::clamp (

const ulonglong3 & v,
const ulonglong3 & a,
const ulonglong3 & b)

clamp

6.1.3.62 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::clamp (

const ulonglong4 & v,
const unsigned long long a,
const unsigned long long b)

clamp

6.1.3.63 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::clamp (

const ulonglong4 & v,
const ulonglong4 & a,
const ulonglong4 & b)

clamp

6.1.3.64 OPTIXU_INLINE float optix::copysignf (

const float dst,
const float src)

copy sign-bit from src value to dst value

6.1.3.65 OPTIXU_INLINE RT_HOSTDEVICE void optix::cosine_sample_hemisphere (

const float u1,
const float u2,
float3 & p)

6.1.3.66 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::cross (

const float3 & a,
const float3 & b)

cross product

6.1.3.67 OPTIXU_INLINE RT_HOSTDEVICE float optix::dot (
 const float2 & a,
 const float2 & b)

dot product

6.1.3.68 OPTIXU_INLINE RT_HOSTDEVICE float optix::dot (
 const float3 & a,
 const float3 & b)

dot product

6.1.3.69 OPTIXU_INLINE RT_HOSTDEVICE float optix::dot (
 const float4 & a,
 const float4 & b)

dot product

6.1.3.70 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::expf (
 const float2 & v)

exp

6.1.3.71 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::expf (
 const float3 & v)

exp

6.1.3.72 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::expf (
 const float4 & v)

exp

6.1.3.73 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::faceforward (
 const float2 & n,
 const float2 & i,
 const float2 & nref)

Faceforward Returns N if dot(i, nref) > 0; else -N; Typical usage is N = faceforward(N, -ray.dir, N); Note that this is opposite of what faceforward does in Cg and GLSL.

6.1.3.74 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::faceforward (
 const float3 & n,
 const float3 & i,
 const float3 & nref)

Faceforward Returns N if dot(i, nref) > 0; else -N; Typical usage is N = faceforward(N, -ray.dir, N); Note that this is opposite of what faceforward does in Cg and GLSL.

6.1.3.75 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::faceforward (
 const float4 & n,
 const float4 & i,
 const float4 & nref)

Faceforward Returns N if dot(i, nref) > 0; else -N; Typical usage is N = faceforward(N, -ray.dir, N); Note that this is opposite of what faceforward does in Cg and GLSL.

6.1.3.76 __device__ int4 optix::float4AsInt4 (
 float4 f4) [inline]

6.1.3.77 __device__ uint4 optix::float4AsUInt4 (
 float4 f4) [inline]

6.1.3.78 OPTIXU_INLINE int optix::float_as_int (
 const float f)

Bit preserving casting function.

6.1.3.79 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::floor (
 const float2 & v)

floor

6.1.3.80 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::floor (
 const float3 & v)

floor

6.1.3.81 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::floor (
 const float4 & v)

floor

6.1.3.82 OPTIXU_INLINE float optix::fmaxf (
 const float a,
 const float b)

6.1.3.83 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::fmaxf (
 const float2 & a,
 const float2 & b)

max

6.1.3.84 OPTIXU_INLINE RT_HOSTDEVICE float optix::fmaxf (
 const float2 & a)

max

6.1.3.85 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::fmaxf (
 const float3 & a,
 const float3 & b)

max

6.1.3.86 OPTIXU_INLINE RT_HOSTDEVICE float optix::fmaxf (
 const float3 & a)

max

6.1.3.87 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::fmaxf (
 const float4 & a,
 const float4 & b)

max

6.1.3.88 OPTIXU_INLINE RT_HOSTDEVICE float optix::fmaxf (
 const float4 & a)

max

6.1.3.89 OPTIXU_INLINE float optix::fminf (
 const float a,
 const float b)

6.1.3.90 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::fminf (
 const float2 & a,
 const float2 & b)

min

6.1.3.91 OPTIXU_INLINE RT_HOSTDEVICE float optix::fminf (
 const float2 & a)

min

6.1.3.92 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::fminf (
 const float3 & a,
 const float3 & b)

min

6.1.3.93 OPTIXU_INLINE RT_HOSTDEVICE float optix::fminf (
 const float3 & a)

min

6.1.3.94 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::fminf (

```
    const float4 & a,  
    const float4 & b )
```

min

6.1.3.95 OPTIXU_INLINE RT_HOSTDEVICE float optix::fminf (
const float4 & a)

min

6.1.3.96 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)

Schlick approximation of Fresnel reflectance.

6.1.3.97 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::fresnel_schlick (
const float cos_theta,
const float exponent,
const float3 & minimum,
const float3 & maximum)

6.1.3.98 OPTIXU_INLINE RT_HOSTDEVICE float optix::getByIndex (
const float1 & v,
int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.99 OPTIXU_INLINE RT_HOSTDEVICE float optix::getByIndex (
const float2 & v,
int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.100 OPTIXU_INLINE RT_HOSTDEVICE float optix::getByIndex (
const float3 & v,
int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.101 OPTIXU_INLINE RT_HOSTDEVICE float optix::getByIndex (
const float4 & v,
int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.102 OPTIXU_INLINE RT_HOSTDEVICE int optix::getByIndex (
 const int1 & v,
 int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.103 OPTIXU_INLINE RT_HOSTDEVICE int optix::getByIndex (
 const int2 & v,
 int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.104 OPTIXU_INLINE RT_HOSTDEVICE int optix::getByIndex (
 const int3 & v,
 int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.105 OPTIXU_INLINE RT_HOSTDEVICE int optix::getByIndex (
 const int4 & v,
 int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.106 OPTIXU_INLINE RT_HOSTDEVICE unsigned int optix::getByIndex (
 const uint1 & v,
 unsigned int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.107 OPTIXU_INLINE RT_HOSTDEVICE unsigned int optix::getByIndex (
 const uint2 & v,
 unsigned int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.108 OPTIXU_INLINE RT_HOSTDEVICE unsigned int optix::getByIndex (
 const uint3 & v,
 unsigned int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.109 OPTIXU_INLINE RT_HOSTDEVICE unsigned int optix::getByIndex (
 const uint4 & v,
 unsigned int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.110 OPTIXU_INLINE RT_HOSTDEVICE long long optix::getByIndex (
 const longlong1 & v,
 int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.111 OPTIXU_INLINE RT_HOSTDEVICE long long optix::getByIndex (
 const longlong2 & v,
 int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.112 OPTIXU_INLINE RT_HOSTDEVICE long long optix::getByIndex (
 const longlong3 & v,
 int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.113 OPTIXU_INLINE RT_HOSTDEVICE long long optix::getByIndex (
 const longlong4 & v,
 int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.114 OPTIXU_INLINE RT_HOSTDEVICE unsigned long long optix::getByIndex (
 const ulonglong1 & v,
 unsigned int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.115 OPTIXU_INLINE RT_HOSTDEVICE unsigned long long optix::getByIndex (
 const ulonglong2 & v,
 unsigned int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.116 OPTIXU_INLINE RT_HOSTDEVICE unsigned long long optix::getByIndex (
 const ulonglong3 & v,
 unsigned int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.117 OPTIXU_INLINE RT_HOSTDEVICE unsigned long long optix::getByIndex (
 const ulonglong4 & v,
 unsigned int i)

If used on the device, this could place the the 'v' in local memory.

6.1.3.118 OPTIXU_INLINE float optix::int_as_float (
int *i*)

Bit preserving casting function.

6.1.3.119 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*)

Intersect ray with CCW wound triangle.

Returns non-normalize normal vector.

6.1.3.120 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*)

Branchless intersection avoids divergence.

6.1.3.121 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*)

Intersection with early exit.

6.1.3.122 OPTIXU_INLINE RT_HOSTDEVICE float optix::length (

const float2 & v)

length

6.1.3.123 OPTIXU_INLINE RT_HOSTDEVICE float optix::length (
const float3 & v)

length

6.1.3.124 OPTIXU_INLINE RT_HOSTDEVICE float optix::length (
const float4 & r)

length

6.1.3.125 OPTIXU_INLINE RT_HOSTDEVICE float optix::lerp (
const float a,
const float b,
const float t)

lerp

6.1.3.126 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::lerp (
const float2 & a,
const float2 & b,
const float t)

lerp

6.1.3.127 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::lerp (
const float3 & a,
const float3 & b,
const float t)

lerp

6.1.3.128 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::lerp (
const float4 & a,
const float4 & b,
const float t)

lerp

6.1.3.129 OPTIXU_INLINE RT_HOSTDEVICE float optix::luminance (
const float3 & rgb)

Calculate the NTSC luminance value of an rgb triple.

6.1.3.130 OPTIXU_INLINE RT_HOSTDEVICE float optix::luminanceCIE (

const float3 & *rgb*)

Calculate the CIE luminance value of an rgb triple.

6.1.3.131 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::make_float2 (
const float *s*)

additional constructors

6.1.3.132 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::make_float2 (
const int2 & *a*)

additional constructors

6.1.3.133 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::make_float2 (
const uint2 & *a*)

additional constructors

6.1.3.134 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::make_float2 (
const float3 & *v0*)

Narrowing functions.

6.1.3.135 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::make_float2 (
const float4 & *v0*)

Narrowing functions.

6.1.3.136 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::make_float3 (
const float *s*)

additional constructors

6.1.3.137 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::make_float3 (
const float2 & *a*)

additional constructors

6.1.3.138 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::make_float3 (
const int3 & *a*)

additional constructors

6.1.3.139 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::make_float3 (
const uint3 & *a*)

additional constructors

6.1.3.140 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::make_float3 (

```
    const float4 & v0 )
```

Narrowing functions.

6.1.3.141 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::make_float3 (
 const float2 & v0,
 const float v1)

Assemble functions from smaller vectors.

6.1.3.142 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::make_float3 (
 const float v0,
 const float2 & v1)

Assemble functions from smaller vectors.

6.1.3.143 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::make_float4 (
 const float s)

additional constructors

6.1.3.144 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::make_float4 (
 const float3 & a)

additional constructors

6.1.3.145 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::make_float4 (
 const int4 & a)

additional constructors

6.1.3.146 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::make_float4 (
 const uint4 & a)

additional constructors

6.1.3.147 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::make_float4 (
 const float v0,
 const float v1,
 const float2 & v2)

Assemble functions from smaller vectors.

6.1.3.148 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::make_float4 (
 const float v0,
 const float2 & v1,
 const float v2)

Assemble functions from smaller vectors.

6.1.3.149 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::make_float4 (
 const float2 & v0,
 const float v1,
 const float v2)

Assemble functions from smaller vectors.

6.1.3.150 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::make_float4 (
 const float v0,
 const float3 & v1)

Assemble functions from smaller vectors.

6.1.3.151 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::make_float4 (
 const float3 & v0,
 const float v1)

Assemble functions from smaller vectors.

6.1.3.152 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::make_float4 (
 const float2 & v0,
 const float2 & v1)

Assemble functions from smaller vectors.

6.1.3.153 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::make_int2 (
 const int s)

additional constructors

6.1.3.154 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::make_int2 (
 const float2 & a)

additional constructors

6.1.3.155 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::make_int2 (
 const int3 & v0)

Narrowing functions.

6.1.3.156 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::make_int2 (
 const int4 & v0)

Narrowing functions.

6.1.3.157 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::make_int3 (
 const int s)

additional constructors

6.1.3.158 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::make_int3 (
const float3 & a)

additional constructors

6.1.3.159 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::make_int3 (
const int4 & v0)

Narrowing functions.

6.1.3.160 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::make_int3 (
const int v0,
const int2 & v1)

Assemble functions from smaller vectors.

6.1.3.161 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::make_int3 (
const int2 & v0,
const int v1)

Assemble functions from smaller vectors.

6.1.3.162 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::make_int4 (
const int s)

additional constructors

6.1.3.163 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::make_int4 (
const float4 & a)

additional constructors

6.1.3.164 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::make_int4 (
const int v0,
const int v1,
const int2 & v2)

Assemble functions from smaller vectors.

6.1.3.165 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::make_int4 (
const int v0,
const int2 & v1,
const int v2)

Assemble functions from smaller vectors.

6.1.3.166 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::make_int4 (
const int2 & v0,

```
    const int v1,  
    const int v2 )
```

Assemble functions from smaller vectors.

6.1.3.167 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::make_int4 (
 const int v0,
 const int3 & v1)

Assemble functions from smaller vectors.

6.1.3.168 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::make_int4 (
 const int3 & v0,
 const int v1)

Assemble functions from smaller vectors.

6.1.3.169 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::make_int4 (
 const int2 & v0,
 const int2 & v1)

Assemble functions from smaller vectors.

6.1.3.170 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::make_longlong2 (
 const long long s)

additional constructors

6.1.3.171 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::make_longlong2 (
 const float2 & a)

additional constructors

6.1.3.172 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::make_longlong2 (
 const longlong3 & v0)

Narrowing functions.

6.1.3.173 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::make_longlong2 (
 const longlong4 & v0)

Narrowing functions.

6.1.3.174 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::make_longlong3 (
 const long long s)

additional constructors

6.1.3.175 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::make_longlong3 (

```
    const float3 & a )
```

additional constructors

6.1.3.176 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::make_longlong3 (
const longlong4 & v0)

Narrowing functions.

6.1.3.177 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::make_longlong3 (
const long long v0,
const longlong2 & v1)

Assemble functions from smaller vectors.

6.1.3.178 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::make_longlong3 (
const longlong2 & v0,
const long long v1)

Assemble functions from smaller vectors.

6.1.3.179 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::make_longlong4 (
const long long s)

additional constructors

6.1.3.180 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::make_longlong4 (
const float4 & a)

additional constructors

6.1.3.181 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::make_longlong4 (
const long long v0,
const long long v1,
const longlong2 & v2)

Assemble functions from smaller vectors.

6.1.3.182 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::make_longlong4 (
const long long v0,
const longlong2 & v1,
const long long v2)

Assemble functions from smaller vectors.

6.1.3.183 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::make_longlong4 (
const longlong2 & v0,
const long long v1,

```
    const long long v2 )
```

Assemble functions from smaller vectors.

6.1.3.184 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::make_longlong4 (
 const long long v0,
 const longlong3 & v1)

Assemble functions from smaller vectors.

6.1.3.185 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::make_longlong4 (
 const longlong3 & v0,
 const long long v1)

Assemble functions from smaller vectors.

6.1.3.186 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::make_longlong4 (
 const longlong2 & v0,
 const longlong2 & v1)

Assemble functions from smaller vectors.

6.1.3.187 OPTIXU_INLINE RT_HOSTDEVICE Matrix<3,3> optix::make_matrix3x3 (
 const Matrix< 4, 4 > & matrix)

6.1.3.188 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::make_uint2 (
 const unsigned int s)

additional constructors

6.1.3.189 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::make_uint2 (
 const float2 & a)

additional constructors

6.1.3.190 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::make_uint2 (
 const uint3 & v0)

Narrowing functions.

6.1.3.191 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::make_uint2 (
 const uint4 & v0)

Narrowing functions.

6.1.3.192 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::make_uint3 (
 const unsigned int s)

additional constructors

6.1.3.193 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::make_uint3 (
 const float3 & a)

additional constructors

6.1.3.194 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::make_uint3 (
 const uint4 & v0)

Narrowing functions.

6.1.3.195 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::make_uint3 (
 const unsigned int v0,
 const uint2 & v1)

Assemble functions from smaller vectors.

6.1.3.196 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::make_uint3 (
 const uint2 & v0,
 const unsigned int v1)

Assemble functions from smaller vectors.

6.1.3.197 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::make_uint4 (
 const unsigned int s)

additional constructors

6.1.3.198 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::make_uint4 (
 const float4 & a)

additional constructors

6.1.3.199 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::make_uint4 (
 const unsigned int v0,
 const unsigned int v1,
 const uint2 & v2)

Assemble functions from smaller vectors.

6.1.3.200 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::make_uint4 (
 const unsigned int v0,
 const uint2 & v1,
 const unsigned int v2)

Assemble functions from smaller vectors.

6.1.3.201 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::make_uint4 (
 const uint2 & v0,

```
    const unsigned int v1,  
    const unsigned int v2 )
```

Assemble functions from smaller vectors.

6.1.3.202 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::make_uint4 (
 const unsigned int v0,
 const uint3 & v1)

Assemble functions from smaller vectors.

6.1.3.203 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::make_uint4 (
 const uint3 & v0,
 const unsigned int v1)

Assemble functions from smaller vectors.

6.1.3.204 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::make_uint4 (
 const uint2 & v0,
 const uint2 & v1)

Assemble functions from smaller vectors.

6.1.3.205 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::make_ulonglong2 (
 const unsigned long long s)

additional constructors

6.1.3.206 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::make_ulonglong2 (
 const float2 & a)

additional constructors

6.1.3.207 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::make_ulonglong2 (
 const ulonglong3 & v0)

Narrowing functions.

6.1.3.208 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::make_ulonglong2 (
 const ulonglong4 & v0)

Narrowing functions.

6.1.3.209 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::make_ulonglong3 (
 const unsigned long long s)

additional constructors

6.1.3.210 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::make_ulonglong3 (

```
    const float3 & a )
```

additional constructors

6.1.3.211 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::make_ulonglong3 (
const ulonglong4 & v0)

Narrowing functions.

6.1.3.212 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::make_ulonglong3 (
const unsigned long long v0,
const ulonglong2 & v1)

Assemble functions from smaller vectors.

6.1.3.213 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::make_ulonglong3 (
const ulonglong2 & v0,
const unsigned long long v1)

Assemble functions from smaller vectors.

6.1.3.214 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::make_ulonglong4 (
const unsigned long long s)

additional constructors

6.1.3.215 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::make_ulonglong4 (
const float4 & a)

additional constructors

6.1.3.216 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::make_ulonglong4 (
const unsigned long long v0,
const unsigned long long v1,
const ulonglong2 & v2)

Assemble functions from smaller vectors.

6.1.3.217 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::make_ulonglong4 (
const unsigned long long v0,
const ulonglong2 & v1,
const unsigned long long v2)

Assemble functions from smaller vectors.

6.1.3.218 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::make_ulonglong4 (
const ulonglong2 & v0,
const unsigned long long v1,

```
    const unsigned long long v2 )
```

Assemble functions from smaller vectors.

6.1.3.219 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::make_ulonglong4 (
 const unsigned long long v0,
 const ulonglong3 & v1)

Assemble functions from smaller vectors.

6.1.3.220 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::make_ulonglong4 (
 const ulonglong3 & v0,
 const unsigned long long v1)

Assemble functions from smaller vectors.

6.1.3.221 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::make_ulonglong4 (
 const ulonglong2 & v0,
 const ulonglong2 & v1)

Assemble functions from smaller vectors.

6.1.3.222 OPTIXU_INLINE int optix::max (
 int a,
 int b)

6.1.3.223 OPTIXU_INLINE long long optix::max (
 long long a,
 long long b)

6.1.3.224 OPTIXU_INLINE unsigned int optix::max (
 unsigned int a,
 unsigned int b)

6.1.3.225 OPTIXU_INLINE unsigned long long optix::max (
 unsigned long long a,
 unsigned long long b)

6.1.3.226 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::max (
 const int2 & a,
 const int2 & b)

max

6.1.3.227 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::max (
 const int3 & a,

const int3 & b)

max

6.1.3.228 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::max (
 const int4 & a,
 const int4 & b)

max

6.1.3.229 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::max (
 const uint2 & a,
 const uint2 & b)

max

6.1.3.230 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::max (
 const uint3 & a,
 const uint3 & b)

max

6.1.3.231 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::max (
 const uint4 & a,
 const uint4 & b)

max

6.1.3.232 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::max (
 const longlong2 & a,
 const longlong2 & b)

max

6.1.3.233 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::max (
 const longlong3 & a,
 const longlong3 & b)

max

6.1.3.234 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::max (
 const longlong4 & a,
 const longlong4 & b)

max

6.1.3.235 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::max (
 const ulonglong2 & a,

const ulonglong2 & b)

max

6.1.3.236 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::max (

const ulonglong3 & a,

const ulonglong3 & b)

max

6.1.3.237 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::max (

const ulonglong4 & a,

const ulonglong4 & b)

max

6.1.3.238 OPTIXU_INLINE int optix::min (

int a,

int b)

6.1.3.239 OPTIXU_INLINE long long optix::min (

long long a,

long long b)

6.1.3.240 OPTIXU_INLINE unsigned int optix::min (

unsigned int a,

unsigned int b)

6.1.3.241 OPTIXU_INLINE unsigned long long optix::min (

unsigned long long a,

unsigned long long b)

6.1.3.242 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::min (

const int2 & a,

const int2 & b)

min

6.1.3.243 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::min (

const int3 & a,

const int3 & b)

min

6.1.3.244 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::min (

const int4 & a,

const int4 & *b*)

min

6.1.3.245 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::min (

const uint2 & *a*,

const uint2 & *b*)

min

6.1.3.246 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::min (

const uint3 & *a*,

const uint3 & *b*)

min

6.1.3.247 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::min (

const uint4 & *a*,

const uint4 & *b*)

min

6.1.3.248 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::min (

const longlong2 & *a*,

const longlong2 & *b*)

min

6.1.3.249 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::min (

const longlong3 & *a*,

const longlong3 & *b*)

min

6.1.3.250 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::min (

const longlong4 & *a*,

const longlong4 & *b*)

min

6.1.3.251 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::min (

const ulonglong2 & *a*,

const ulonglong2 & *b*)

min

6.1.3.252 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::min (

const ulonglong3 & *a*,

const ulonglong3 & b)

min

6.1.3.253 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::min (

const ulonglong4 & a,

const ulonglong4 & b)

min

6.1.3.254 OPTIXU_INLINE RT_HOSTDEVICE Quaternion optix::nlerp (

const Quaternion & quat0,

const Quaternion & quat1,

float t)

6.1.3.255 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::normalize (

const float2 & v)

normalize

6.1.3.256 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::normalize (

const float3 & v)

normalize

6.1.3.257 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::normalize (

const float4 & v)

normalize

6.1.3.258 RT_HOSTDEVICE bool optix::operator!= (

const Matrix< M, N > & m1,

const Matrix< M, N > & m2)

6.1.3.259 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (

const int2 & a,

const int2 & b)

equality

6.1.3.260 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (

const int3 & a,

const int3 & b)

equality

6.1.3.261 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (

const int4 & a,

const int4 & b)

equality

6.1.3.262 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (

const uint2 & a,

const uint2 & b)

equality

6.1.3.263 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (

const uint3 & a,

const uint3 & b)

equality

6.1.3.264 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (

const uint4 & a,

const uint4 & b)

equality

6.1.3.265 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (

const longlong2 & a,

const longlong2 & b)

equality

6.1.3.266 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (

const longlong3 & a,

const longlong3 & b)

equality

6.1.3.267 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (

const longlong4 & a,

const longlong4 & b)

equality

6.1.3.268 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (

const ulonglong2 & a,

const ulonglong2 & b)

equality

6.1.3.269 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (

const ulonglong3 & a,

```
    const ulonglong3 & b )
```

equality

6.1.3.270 **OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!= (**

```
    const ulonglong4 & a,
```

```
    const ulonglong4 & b )
```

equality

6.1.3.271 **RT_HOSTDEVICE Matrix< M, N > optix::operator* (**

```
    const Matrix< M, N > & m,
```

```
    float f )
```

6.1.3.272 **RT_HOSTDEVICE Matrix< M, N > optix::operator* (**

```
    float f,
```

```
    const Matrix< M, N > & m )
```

6.1.3.273 **RT_HOSTDEVICE Matrix< M, N >::floatM optix::operator* (**

```
    const Matrix< M, N > & m,
```

```
    const typename Matrix< M, N >::floatN & v )
```

6.1.3.274 **RT_HOSTDEVICE Matrix< M, N >::floatN optix::operator* (**

```
    const typename Matrix< M, N >::floatM & v,
```

```
    const Matrix< M, N > & m )
```

6.1.3.275 **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 )
```

6.1.3.276 **template<unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator***

```
(
```

```
    const Matrix< 2, N > & m,
```

```
    const typename Matrix< 2, N >::floatN & vec )
```

6.1.3.277 **template<unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator***

```
(
```

```
    const Matrix< 3, N > & m,
```

```
    const typename Matrix< 3, N >::floatN & vec )
```

6.1.3.278 **template<unsigned int N> OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator***

```
(
```

```
    const Matrix< 4, N > & m,
```

```
    const typename Matrix< 4, N >::floatN & vec )
```

6.1.3.279 **OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator*** (
 const Matrix< 4, 4 > & m,
 const float4 & vec)

6.1.3.280 **OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator*** (
 const Quaternion & quat,
 const float3 & v)

6.1.3.281 **OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator*** (
 const Quaternion & quat,
 const float4 & v)

6.1.3.282 **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)

6.1.3.283 **OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator*** (
 const float2 & a,
 const float2 & b)

multiply

6.1.3.284 **OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator*** (
 const float2 & a,
 const float s)

multiply

6.1.3.285 **OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator*** (
 const float s,
 const float2 & a)

multiply

6.1.3.286 **OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator*** (
 const float3 & a,
 const float3 & b)

multiply

6.1.3.287 **OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator*** (
 const float3 & a,

const float s)

multiply

6.1.3.288 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator* (
 const float s,
 const float3 & a)

multiply

6.1.3.289 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator* (
 const float4 & a,
 const float4 & s)

multiply

6.1.3.290 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator* (
 const float4 & a,
 const float s)

multiply

6.1.3.291 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator* (
 const float s,
 const float4 & a)

multiply

6.1.3.292 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::operator* (
 const int2 & a,
 const int2 & b)

multiply

6.1.3.293 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::operator* (
 const int2 & a,
 const int s)

multiply

6.1.3.294 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::operator* (
 const int s,
 const int2 & a)

multiply

6.1.3.295 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::operator* (
 const int3 & a,

const int3 & b)

multiply

6.1.3.296 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::operator* (

const int3 & a,

const int s)

multiply

6.1.3.297 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::operator* (

const int s,

const int3 & a)

multiply

6.1.3.298 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::operator* (

const int4 & a,

const int4 & b)

multiply

6.1.3.299 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::operator* (

const int4 & a,

const int s)

multiply

6.1.3.300 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::operator* (

const int s,

const int4 & a)

multiply

6.1.3.301 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::operator* (

const uint2 & a,

const uint2 & b)

multiply

6.1.3.302 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::operator* (

const uint2 & a,

const unsigned int s)

multiply

6.1.3.303 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::operator* (

const unsigned int s,

const uint2 & a)

multiply

6.1.3.304 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::operator* (

const uint3 & a,

const uint3 & b)

multiply

6.1.3.305 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::operator* (

const uint3 & a,

const unsigned int s)

multiply

6.1.3.306 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::operator* (

const unsigned int s,

const uint3 & a)

multiply

6.1.3.307 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::operator* (

const uint4 & a,

const uint4 & b)

multiply

6.1.3.308 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::operator* (

const uint4 & a,

const unsigned int s)

multiply

6.1.3.309 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::operator* (

const unsigned int s,

const uint4 & a)

multiply

6.1.3.310 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::operator* (

const longlong2 & a,

const longlong2 & b)

multiply

6.1.3.311 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::operator* (

const longlong2 & a,

const long long s)

multiply

6.1.3.312 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::operator* (
const long long s,
const longlong2 & a)

multiply

6.1.3.313 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::operator* (
const longlong3 & a,
const longlong3 & b)

multiply

6.1.3.314 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::operator* (
const longlong3 & a,
const long long s)

multiply

6.1.3.315 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::operator* (
const long long s,
const longlong3 & a)

multiply

6.1.3.316 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::operator* (
const longlong4 & a,
const longlong4 & b)

multiply

6.1.3.317 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::operator* (
const longlong4 & a,
const long long s)

multiply

6.1.3.318 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::operator* (
const long long s,
const longlong4 & a)

multiply

6.1.3.319 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::operator* (
const ulonglong2 & a,

const ulonglong2 & b)

multiply

6.1.3.320 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::operator* (
const ulonglong2 & a,
const unsigned long long s)

multiply

6.1.3.321 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::operator* (
const unsigned long long s,
const ulonglong2 & a)

multiply

6.1.3.322 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::operator* (
const ulonglong3 & a,
const ulonglong3 & b)

multiply

6.1.3.323 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::operator* (
const ulonglong3 & a,
const unsigned long long s)

multiply

6.1.3.324 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::operator* (
const unsigned long long s,
const ulonglong3 & a)

multiply

6.1.3.325 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::operator* (
const ulonglong4 & a,
const ulonglong4 & b)

multiply

6.1.3.326 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::operator* (
const ulonglong4 & a,
const unsigned long long s)

multiply

6.1.3.327 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::operator* (
const unsigned long long s,

const ulonglong4 & a)

multiply

**6.1.3.328 template<unsigned int M> OPTIXU_INLINE RT_HOSTDEVICE Matrix<M,M>& optix::operator*=(
Matrix< M, M > & m1,
const Matrix< M, M > & m2)**

**6.1.3.329 RT_HOSTDEVICE Matrix< M, N > & optix::operator*=(
Matrix< M, N > & m1,
float f)**

**6.1.3.330 template<unsigned int M> RT_HOSTDEVICE Matrix<M,M>& optix::operator*=(
Matrix< M, M > & m1,
const Matrix< M, M > & m2)**

**6.1.3.331 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
float2 & a,
const float2 & s)**

multiply

**6.1.3.332 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
float2 & a,
const float s)**

multiply

**6.1.3.333 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
float3 & a,
const float3 & s)**

multiply

**6.1.3.334 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
float3 & a,
const float s)**

multiply

**6.1.3.335 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
float4 & a,
const float4 & s)**

multiply

6.1.3.336 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(

```
float4 & a,  
const float s )
```

multiply

6.1.3.337 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
int2 & a,
const int s)

multiply

6.1.3.338 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
int3 & a,
const int s)

multiply

6.1.3.339 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
int4 & a,
const int s)

multiply

6.1.3.340 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
uint2 & a,
const unsigned int s)

multiply

6.1.3.341 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
uint3 & a,
const unsigned int s)

multiply

6.1.3.342 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
uint4 & a,
const unsigned int s)

multiply

6.1.3.343 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
longlong2 & a,
const long long s)

multiply

6.1.3.344 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(

```
longlong3 & a,  
const long long s )
```

multiply

6.1.3.345 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
longlong4 & a,
const long long s)

multiply

6.1.3.346 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
ulonglong2 & a,
const unsigned long long s)

multiply

6.1.3.347 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
ulonglong3 & a,
const unsigned long long s)

multiply

6.1.3.348 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator*=(
ulonglong4 & a,
const unsigned long long s)

multiply

6.1.3.349 RT_HOSTDEVICE Matrix< M, N > optix::operator+ (
const Matrix< M, N > & m1,
const Matrix< M, N > & m2)

6.1.3.350 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator+ (
const float2 & a,
const float2 & b)

add

6.1.3.351 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator+ (
const float2 & a,
const float b)

add

6.1.3.352 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator+ (
const float a,

```
    const float2 & b )
```

add

6.1.3.353 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator+ (

```
    const float3 & a,
```

```
    const float3 & b )
```

add

6.1.3.354 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator+ (

```
    const float3 & a,
```

```
    const float b )
```

add

6.1.3.355 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator+ (

```
    const float a,
```

```
    const float3 & b )
```

add

6.1.3.356 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator+ (

```
    const float4 & a,
```

```
    const float4 & b )
```

add

6.1.3.357 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator+ (

```
    const float4 & a,
```

```
    const float b )
```

add

6.1.3.358 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator+ (

```
    const float a,
```

```
    const float4 & b )
```

add

6.1.3.359 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::operator+ (

```
    const int2 & a,
```

```
    const int2 & b )
```

add

6.1.3.360 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::operator+ (

```
    const int3 & a,
```

const int3 & b)

add

6.1.3.361 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::operator+ (

const int4 & a,

const int4 & b)

add

6.1.3.362 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::operator+ (

const uint2 & a,

const uint2 & b)

add

6.1.3.363 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::operator+ (

const uint3 & a,

const uint3 & b)

add

6.1.3.364 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::operator+ (

const uint4 & a,

const uint4 & b)

add

6.1.3.365 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::operator+ (

const longlong2 & a,

const longlong2 & b)

add

6.1.3.366 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::operator+ (

const longlong3 & a,

const longlong3 & b)

add

6.1.3.367 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::operator+ (

const longlong4 & a,

const longlong4 & b)

add

6.1.3.368 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::operator+ (

const ulonglong2 & a,

const ulonglong2 & b)

add

6.1.3.369 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::operator+ (

const ulonglong3 & a,

const ulonglong3 & b)

add

6.1.3.370 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::operator+ (

const ulonglong4 & a,

const ulonglong4 & b)

add

6.1.3.371 RT_HOSTDEVICE Matrix< M, N > & optix::operator+= (

Matrix< M, N > & m1,

const Matrix< M, N > & m2)

6.1.3.372 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (

float2 & a,

const float2 & b)

add

6.1.3.373 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (

float3 & a,

const float3 & b)

add

6.1.3.374 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (

float4 & a,

const float4 & b)

add

6.1.3.375 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (

int2 & a,

const int2 & b)

add

6.1.3.376 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (

int3 & a,

const int3 & *b*)

add

6.1.3.377 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (
int4 & *a*,
const int4 & *b*)

add

6.1.3.378 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (
uint2 & *a*,
const uint2 & *b*)

add

6.1.3.379 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (
uint3 & *a*,
const uint3 & *b*)

add

6.1.3.380 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (
uint4 & *a*,
const uint4 & *b*)

add

6.1.3.381 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (
longlong2 & *a*,
const longlong2 & *b*)

add

6.1.3.382 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (
longlong3 & *a*,
const longlong3 & *b*)

add

6.1.3.383 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (
longlong4 & *a*,
const longlong4 & *b*)

add

6.1.3.384 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (
ulonglong2 & *a*,

const ulonglong2 & b)

add

6.1.3.385 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (
 ulonglong3 & a,
 const ulonglong3 & b)

add

6.1.3.386 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator+= (
 ulonglong4 & a,
 const ulonglong4 & b)

add

6.1.3.387 RT_HOSTDEVICE Matrix< M, N > optix::operator- (
 const Matrix< M, N > & m1,
 const Matrix< M, N > & m2)

6.1.3.388 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator- (
 const float2 & a)

negate

6.1.3.389 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator- (
 const float2 & a,
 const float2 & b)

subtract

6.1.3.390 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator- (
 const float2 & a,
 const float b)

subtract

6.1.3.391 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator- (
 const float a,
 const float2 & b)

subtract

6.1.3.392 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator- (
 const float3 & a)

negate

6.1.3.393 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator- (

```
    const float3 & a,  
    const float3 & b )
```

subtract

6.1.3.394 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator- (
 const float3 & a,
 const float b)

subtract

6.1.3.395 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator- (
 const float a,
 const float3 & b)

subtract

6.1.3.396 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator- (
 const float4 & a)

negate

6.1.3.397 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator- (
 const float4 & a,
 const float4 & b)

subtract

6.1.3.398 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator- (
 const float4 & a,
 const float b)

subtract

6.1.3.399 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator- (
 const float a,
 const float4 & b)

subtract

6.1.3.400 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::operator- (
 const int2 & a)

negate

6.1.3.401 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::operator- (
 const int2 & a,

const int2 & b)

subtract

6.1.3.402 OPTIXU_INLINE RT_HOSTDEVICE int2 optix::operator- (

const int2 & a,

const int b)

subtract

6.1.3.403 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::operator- (

const int3 & a)

negate

6.1.3.404 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::operator- (

const int3 & a,

const int3 & b)

subtract

6.1.3.405 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::operator- (

const int4 & a)

negate

6.1.3.406 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::operator- (

const int4 & a,

const int4 & b)

subtract

6.1.3.407 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::operator- (

const uint2 & a,

const uint2 & b)

subtract

6.1.3.408 OPTIXU_INLINE RT_HOSTDEVICE uint2 optix::operator- (

const uint2 & a,

const unsigned int b)

subtract

6.1.3.409 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::operator- (

const uint3 & a,

const uint3 & b)

subtract

6.1.3.410 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::operator- (
 const uint4 & a,
 const uint4 & b)

subtract

6.1.3.411 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::operator- (
 const longlong2 & a)

negate

6.1.3.412 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::operator- (
 const longlong2 & a,
 const longlong2 & b)

subtract

6.1.3.413 OPTIXU_INLINE RT_HOSTDEVICE longlong2 optix::operator- (
 const longlong2 & a,
 const long long b)

subtract

6.1.3.414 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::operator- (
 const longlong3 & a)

negate

6.1.3.415 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::operator- (
 const longlong3 & a,
 const longlong3 & b)

subtract

6.1.3.416 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::operator- (
 const longlong4 & a)

negate

6.1.3.417 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::operator- (
 const longlong4 & a,
 const longlong4 & b)

subtract

6.1.3.418 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::operator- (
 const ulonglong2 & a,

const ulonglong2 & *b*)

subtract

6.1.3.419 OPTIXU_INLINE RT_HOSTDEVICE ulonglong2 optix::operator- (
const ulonglong2 & *a*,
const unsigned long long *b*)

subtract

6.1.3.420 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::operator- (
const ulonglong3 & *a*,
const ulonglong3 & *b*)

subtract

6.1.3.421 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::operator- (
const ulonglong4 & *a*,
const ulonglong4 & *b*)

subtract

6.1.3.422 RT_HOSTDEVICE Matrix< M, N > & optix::operator-= (
Matrix< M, N > & *m1*,
const Matrix< M, N > & *m2*)

6.1.3.423 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
float2 & *a*,
const float2 & *b*)

subtract

6.1.3.424 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
float3 & *a*,
const float3 & *b*)

subtract

6.1.3.425 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
float4 & *a*,
const float4 & *b*)

subtract

6.1.3.426 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
int2 & *a*,

const int2 & b)

subtract

6.1.3.427 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
int3 & a,
const int3 & b)

subtract

6.1.3.428 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
int4 & a,
const int4 & b)

subtract

6.1.3.429 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
uint2 & a,
const uint2 & b)

subtract

6.1.3.430 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
uint3 & a,
const uint3 & b)

subtract

6.1.3.431 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
uint4 & a,
const uint4 & b)

subtract

6.1.3.432 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
longlong2 & a,
const longlong2 & b)

subtract

6.1.3.433 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
longlong3 & a,
const longlong3 & b)

subtract

6.1.3.434 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
longlong4 & a,

const longlong4 & b)

subtract

6.1.3.435 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
ullonglong2 & a,
const ullonglong2 & b)

subtract

6.1.3.436 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
ullonglong3 & a,
const ullonglong3 & b)

subtract

6.1.3.437 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator-= (
ullonglong4 & a,
const ullonglong4 & b)

subtract

6.1.3.438 RT_HOSTDEVICE Matrix< M, N > optix::operator/ (
const Matrix< M, N > & m,
float f)

6.1.3.439 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator/ (
const float2 & a,
const float2 & b)

divide

6.1.3.440 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator/ (
const float2 & a,
const float s)

divide

6.1.3.441 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::operator/ (
const float s,
const float2 & a)

divide

6.1.3.442 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator/ (
const float3 & a,

const float3 & b)

divide

6.1.3.443 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator/ (

const float3 & a,

const float s)

divide

6.1.3.444 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::operator/ (

const float s,

const float3 & a)

divide

6.1.3.445 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator/ (

const float4 & a,

const float4 & b)

divide

6.1.3.446 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator/ (

const float4 & a,

const float s)

divide

6.1.3.447 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::operator/ (

const float s,

const float4 & a)

divide

6.1.3.448 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::operator/ (

const int3 & a,

const int3 & b)

divide

6.1.3.449 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::operator/ (

const int3 & a,

const int s)

divide

6.1.3.450 OPTIXU_INLINE RT_HOSTDEVICE int3 optix::operator/ (

const int s,

const int3 & a)

divide

6.1.3.451 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::operator/ (

const int4 & a,

const int4 & b)

divide

6.1.3.452 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::operator/ (

const int4 & a,

const int s)

divide

6.1.3.453 OPTIXU_INLINE RT_HOSTDEVICE int4 optix::operator/ (

const int s,

const int4 & a)

divide

6.1.3.454 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::operator/ (

const uint3 & a,

const uint3 & b)

divide

6.1.3.455 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::operator/ (

const uint3 & a,

const unsigned int s)

divide

6.1.3.456 OPTIXU_INLINE RT_HOSTDEVICE uint3 optix::operator/ (

const unsigned int s,

const uint3 & a)

divide

6.1.3.457 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::operator/ (

const uint4 & a,

const uint4 & b)

divide

6.1.3.458 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::operator/ (

const uint4 & a,

const unsigned int s)

divide

6.1.3.459 OPTIXU_INLINE RT_HOSTDEVICE uint4 optix::operator/ (
const unsigned int s,
const uint4 & a)

divide

6.1.3.460 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::operator/ (
const longlong3 & a,
const longlong3 & b)

divide

6.1.3.461 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::operator/ (
const longlong3 & a,
const long long s)

divide

6.1.3.462 OPTIXU_INLINE RT_HOSTDEVICE longlong3 optix::operator/ (
const long long s,
const longlong3 & a)

divide

6.1.3.463 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::operator/ (
const longlong4 & a,
const longlong4 & b)

divide

6.1.3.464 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::operator/ (
const longlong4 & a,
const long long s)

divide

6.1.3.465 OPTIXU_INLINE RT_HOSTDEVICE longlong4 optix::operator/ (
const long long s,
const longlong4 & a)

divide

6.1.3.466 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::operator/ (
const ulonglong3 & a,

const ulonglong3 & b)

divide

6.1.3.467 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::operator/ (
const ulonglong3 & a,
const unsigned long long s)

divide

6.1.3.468 OPTIXU_INLINE RT_HOSTDEVICE ulonglong3 optix::operator/ (
const unsigned long long s,
const ulonglong3 & a)

divide

6.1.3.469 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::operator/ (
const ulonglong4 & a,
const ulonglong4 & b)

divide

6.1.3.470 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::operator/ (
const ulonglong4 & a,
const unsigned long long s)

divide

6.1.3.471 OPTIXU_INLINE RT_HOSTDEVICE ulonglong4 optix::operator/ (
const unsigned long long s,
const ulonglong4 & a)

divide

6.1.3.472 RT_HOSTDEVICE Matrix< M, N > & optix::operator/= (
Matrix< M, N > & m1,
float f)

6.1.3.473 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator/= (
float2 & a,
const float s)

divide

6.1.3.474 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator/= (
float3 & a,

const float s)

divide

6.1.3.475 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator/= (

float4 & a,
const float s)

divide

6.1.3.476 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator/= (

int3 & a,
const int s)

divide

6.1.3.477 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator/= (

int4 & a,
const int s)

divide

6.1.3.478 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator/= (

uint3 & a,
const unsigned int s)

divide

6.1.3.479 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator/= (

uint4 & a,
const unsigned int s)

divide

6.1.3.480 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator/= (

longlong3 & a,
const long long s)

divide

6.1.3.481 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator/= (

longlong4 & a,
const long long s)

divide

6.1.3.482 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator/= (

ulonglong3 & a,

const unsigned long long s)

divide

6.1.3.483 OPTIXU_INLINE RT_HOSTDEVICE void optix::operator/= (
ulonglong4 & a,
const unsigned long long s)

divide

6.1.3.484 std::ostream& optix::operator<< (
std::ostream & os,
const optix::float4 & v) [inline]

Provide access to stream functionalities with CUDA float vector types.

6.1.3.485 std::ostream& optix::operator<< (
std::ostream & os,
const optix::float3 & v) [inline]

Provide access to stream functionalities with CUDA float vector types.

6.1.3.486 std::ostream& optix::operator<< (
std::ostream & os,
const optix::float2 & v) [inline]

Provide access to stream functionalities with CUDA float vector types.

6.1.3.487 std::ostream& optix::operator<< (
std::ostream & os,
const optix::int4 & v) [inline]

Provide access to stream functionalities with CUDA int vector types.

6.1.3.488 std::ostream& optix::operator<< (
std::ostream & os,
const optix::int3 & v) [inline]

Provide access to stream functionalities with CUDA int vector types.

6.1.3.489 std::ostream& optix::operator<< (
std::ostream & os,
const optix::int2 & v) [inline]

Provide access to stream functionalities with CUDA int vector types.

6.1.3.490 std::ostream& optix::operator<< (
std::ostream & os,

const optix::uint4 & v) [inline]

Provide access to stream functionalities with CUDA uint vector types.

**6.1.3.491 std::ostream& optix::operator<< (std::ostream & os,
const optix::uint3 & v) [inline]**

Provide access to stream functionalities with CUDA uint vector types.

**6.1.3.492 std::ostream& optix::operator<< (std::ostream & os,
const optix::uint2 & v) [inline]**

Provide access to stream functionalities with CUDA uint vector types.

**6.1.3.493 std::ostream& optix::operator<< (std::ostream & os,
const optix::Aabb & aabb) [inline]**

Provide access to stream functionalities with OptiX axis-aligned bounding box type.

**6.1.3.494 template<unsigned int M, unsigned int N> std::ostream& optix::operator<< (std::ostream & os,
const optix::Matrix< M, N > & m) [inline]**

Provide access to stream functionalities with OptiX matrix type.

**6.1.3.495 RT_HOSTDEVICE bool optix::operator== (const Matrix< M, N > & m1,
const Matrix< M, N > & m2)****6.1.3.496 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator== (const int2 & a,
const int2 & b)**

equality

**6.1.3.497 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator== (const int3 & a,
const int3 & b)**

equality

6.1.3.498 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator== (const int4 & a,

const int4 & b)

equality

6.1.3.499 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator==(
const uint2 & a,
const uint2 & b)

equality

6.1.3.500 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator==(
const uint3 & a,
const uint3 & b)

equality

6.1.3.501 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator==(
const uint4 & a,
const uint4 & b)

equality

6.1.3.502 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator==(
const longlong2 & a,
const longlong2 & b)

equality

6.1.3.503 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator==(
const longlong3 & a,
const longlong3 & b)

equality

6.1.3.504 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator==(
const longlong4 & a,
const longlong4 & b)

equality

6.1.3.505 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator==(
const ulonglong2 & a,
const ulonglong2 & b)

equality

6.1.3.506 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator==(
const ulonglong3 & a,

const ulonglong3 & *b*)

equality

**6.1.3.507 OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator==(
const ulonglong4 & *a*,
const ulonglong4 & *b*)**

equality

**6.1.3.508 std::istream& optix::operator>>(
std::istream & *is*,
optix::float4 & *v*) [inline]**

Provide access to stream functionalities with CUDA float vector types.

**6.1.3.509 std::istream& optix::operator>>(
std::istream & *is*,
optix::float3 & *v*) [inline]**

Provide access to stream functionalities with CUDA float vector types.

**6.1.3.510 std::istream& optix::operator>>(
std::istream & *is*,
optix::float2 & *v*) [inline]**

Provide access to stream functionalities with CUDA float vector types.

**6.1.3.511 std::istream& optix::operator>>(
std::istream & *is*,
optix::int4 & *v*) [inline]**

Provide access to stream functionalities with CUDA int vector types.

**6.1.3.512 std::istream& optix::operator>>(
std::istream & *is*,
optix::int3 & *v*) [inline]**

Provide access to stream functionalities with CUDA int vector types.

**6.1.3.513 std::istream& optix::operator>>(
std::istream & *is*,
optix::int2 & *v*) [inline]**

Provide access to stream functionalities with CUDA int vector types.

**6.1.3.514 std::istream& optix::operator>>(
std::istream & *is*,**

optix::uint4 & v) [inline]

Provide access to stream functionalities with CUDA uint vector types.

6.1.3.515 std::istream& optix::operator>>(

std::istream & is,

optix::uint3 & v) [inline]

Provide access to stream functionalities with CUDA uint vector types.

6.1.3.516 std::istream& optix::operator>>(

std::istream & is,

optix::uint2 & v) [inline]

Provide access to stream functionalities with CUDA uint vector types.

6.1.3.517 template<unsigned int M, unsigned int N> std::istream& optix::operator>>(

std::istream & is,

optix::Matrix< M, N > & m) [inline]

Provide access to stream functionalities with OptiX matrix type.

6.1.3.518 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::reflect(

const float2 & i,

const float2 & n)

reflect

6.1.3.519 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::reflect(

const float3 & i,

const float3 & n)

reflect

6.1.3.520 OPTIXU_INLINE RT_HOSTDEVICE float4 optix::reflect(

const float4 & i,

const float4 & n)

reflect

6.1.3.521 OPTIXU_INLINE RT_HOSTDEVICE bool optix::refract(

float3 & r,

const float3 & i,

const float3 & n,

const float ior)

Calculates refraction direction r : refraction vector i : incident vector n : surface normal ior : index of refraction (n2 / n1) returns false in case of total internal reflection, in that case r is initialized to (0,0,0).

-
- ```
6.1.3.522 static __forceinline__ __device__ void* optix::rt_buffer_get (
 void * buffer,
 unsigned int dim,
 unsigned int element_size,
 size_t i0_in,
 size_t i1_in,
 size_t i2_in,
 size_t i3_in) [static]
```
- ```
6.1.3.523 static __forceinline__ __device__ void* optix::rt_buffer_get_id (
    int id,
    unsigned int dim,
    unsigned int element_size,
    size_t i0_in,
    size_t i1_in,
    size_t i2_in,
    size_t i3_in ) [static]
```
- ```
6.1.3.524 static __forceinline__ __device__ size_t4 optix::rt_buffer_get_size (
 const void * buffer,
 unsigned int dim,
 unsigned int element_size) [static]
```
- ```
6.1.3.525 static __forceinline__ __device__ size_t4 optix::rt_buffer_get_size_id (
    int id,
    unsigned int dim,
    unsigned int element_size ) [static]
```
- ```
6.1.3.526 static __forceinline__ __device__ void* optix::rt_callable_program_from_id (
 int id,
 const char * csId = 0) [static]
```
- ```
6.1.3.527 static __forceinline__ __device__ unsigned int optix::rt_get_exception_code ( )
    [static]
```
- ```
6.1.3.528 static __forceinline__ __device__ unsigned int optix::rt_get_primitive_index ()
 [static]
```
- ```
6.1.3.529 static __forceinline__ __device__ void optix::rt_get_transform (
    RTtransformkind kind,
```

```
    float matrix[16] ) [static]

6.1.3.530 static __forceinline__ __device__ float2 optix::rt_get_triangle_barycentrics( )
[static]

6.1.3.531 static __forceinline__ __device__ void optix::rt_ignore_intersection( ) [static]

6.1.3.532 template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T ,
typename Arg3T > class callableProgramId< ReturnT(Arg0T, Arg1T, Arg2T, Arg3T)>
optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS( )

6.1.3.533 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( )

6.1.3.534 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(
)

6.1.3.535 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( )

6.1.3.536 template<typename ReturnT , typename Arg0T , typename Arg1T ,
typename Arg2T > class callableProgramId< ReturnT(Arg0T, Arg1T, Arg2T)>
optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS( )

6.1.3.537 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( )

6.1.3.538 template<typename ReturnT , typename Arg0T , typename
Arg1T > class callableProgramId< ReturnT(Arg0T, Arg1T)>
optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS( )

6.1.3.539 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( )

6.1.3.540 template<typename ReturnT , typename Arg0T > class callableProgramId<
NVIDIA OptiX ReturnT(Arg0T)> optix::RT_INTERNAL_CALLABLE_PROGRAM_DEFS( )

6.1.3.541 template<typename ReturnT , typename Arg0T , typename Arg1T , typename
```

```
    ReturnT )
```

```
6.1.3.544 template<typename ReturnT , typename Arg0T > op-
    tix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS(
        (
            ReturnT ,
            Arg0T )
```

```
6.1.3.545 template<typename ReturnT , typename Arg0T , typename Arg1T >
    optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS(
        ReturnT ,
        Arg0T ,
        Arg1T )
```

```
6.1.3.546 template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T
    > optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS(
        ReturnT ,
        Arg0T ,
        Arg1T ,
        Arg2T )
```

```
6.1.3.547 template<typename ReturnT , typename Arg0T , typename
    Arg1T , typename Arg2T , typename Arg3T > op-
    tix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS(
        (
            ReturnT ,
            Arg0T ,
            Arg1T ,
            Arg2T ,
            Arg3T )
```

```
6.1.3.548 template<typename ReturnT , typename Arg0T , typename
    Arg1T , typename Arg2T , typename Arg3T , typename Arg4T >
    optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS(
        ReturnT ,
        Arg0T ,
        Arg1T ,
        Arg2T ,
        Arg3T ,
        Arg4T )
```

```
6.1.3.549 template<typename ReturnT , typename Arg0T , typename Arg1T ,
    typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T >
    optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS(
```

```
ReturnT,  
Arg0T,  
Arg1T,  
Arg2T,  
Arg3T,  
Arg4T,  
Arg5T )
```

6.1.3.550 `template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T >`
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS(`
 ReturnT,
 Arg0T,
 Arg1T,
 Arg2T,
 Arg3T,
 Arg4T,
 Arg5T,
 Arg6T)

6.1.3.551 `template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T ,`
`typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T , typename Arg7T >`
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS(`
 ReturnT,
 Arg0T,
 Arg1T,
 Arg2T,
 Arg3T,
 Arg4T,
 Arg5T,
 Arg6T,
 Arg7T)

6.1.3.552 `template<typename ReturnT , typename Arg0T , typename Arg1T ,`
`typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T ,`
`typename Arg7T , typename Arg8T >`
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS(`
 ReturnT,
 Arg0T,
 Arg1T,
 Arg2T,
 Arg3T,
 Arg4T,

```
Arg5T,  
Arg6T,  
Arg7T,  
Arg8T )
```

- 6.1.3.553 `template<typename ReturnT , typename Arg0T , typename Arg1T ,
typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T ,
typename Arg6T , typename Arg7T , typename Arg8T , typename Arg9T >
optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS(`
`ReturnT ,
Arg0T ,
Arg1T ,
Arg2T ,
Arg3T ,
Arg4T ,
Arg5T ,
Arg6T ,
Arg7T ,
Arg8T ,
Arg9T)`
- 6.1.3.554 `static __forceinline__ __device__ void optix::rt_intersect_child (`
`unsigned int index) [static]`
- 6.1.3.555 `static __forceinline__ __device__ bool optix::rt_is_triangle_hit() [static]`
- 6.1.3.556 `static __forceinline__ __device__ bool optix::rt_is_triangle_hit_back_face()
[static]`
- 6.1.3.557 `static __forceinline__ __device__ bool optix::rt_is_triangle_hit_front_face()
[static]`
- 6.1.3.558 `static __forceinline__ __device__ bool optix::rt_potential_intersection (`
`float t) [static]`
- 6.1.3.559 `static __forceinline__ __device__ int optix::rt_print_active() [static]`
- 6.1.3.560 `static __forceinline__ __device__ bool optix::rt_report_intersection (`
`unsigned int matlIndex) [static]`
- 6.1.3.561 `static __forceinline__ __device__ void optix::rt_terminate_ray() [static]`
- 6.1.3.562 `static __forceinline__ __device__ float4 optix::rt_texture_get_base_id (`
`int tex,
int dim,`

```
    float x,  
    float y,  
    float z,  
    int layer ) [static]
```

6.1.3.563 static __forceinline__ __device__ float4 optix::rt_texture_get_f_id (
 int *tex*,
 int *dim*,
 float *x*,
 float *y*,
 float *z*,
 float *w*) [static]

6.1.3.564 static __forceinline__ __device__ float4 optix::rt_texture_get_fetch_id (
 int *tex*,
 int *dim*,
 int *x*,
 int *y*,
 int *z*,
 int *w*) [static]

6.1.3.565 static __forceinline__ __device__ float4 optix::rt_texture_get_gather_id (
 int *tex*,
 float *x*,
 float *y*,
 int *comp*) [static]

6.1.3.566 static __forceinline__ __device__ float4 optix::rt_texture_get_grad_id (
 int *tex*,
 int *dim*,
 float *x*,
 float *y*,
 float *z*,
 int *layer*,
 float *dPdx_x*,
 float *dPdx_y*,
 float *dPdx_z*,
 float *dPdy_x*,
 float *dPdy_y*,
 float *dPdy_z*) [static]

6.1.3.567 static __forceinline__ __device__ int4 optix::rt_texture_get_i_id (
 int *tex*,

```
    int dim,
    float x,
    float y,
    float z,
    float w ) [static]

6.1.3.568 static __forceinline__ __device__ float4 optix::rt_texture_get_level_id (
    int tex,
    int dim,
    float x,
    float y,
    float z,
    int layer,
    float level ) [static]

6.1.3.569 static __forceinline__ __device__ uint3 optix::rt_texture_get_size_id (
    int tex ) [static]

6.1.3.570 static __forceinline__ __device__ uint4 optix::rt_texture_get_u_id (
    int tex,
    int dim,
    float x,
    float y,
    float z,
    float w ) [static]

6.1.3.571 static __forceinline__ __device__ void optix::rt_throw (
    unsigned int code ) [static]

6.1.3.572 static __forceinline__ __device__ void optix::rt_trace (
    unsigned int group,
    float3 origin,
    float3 direction,
    unsigned int ray_type,
    float tmin,
    float tmax,
    RTvisibilitymask mask,
    RTrayflags flags,
    void * prd,
    unsigned int prd_size ) [static]

6.1.3.573 static __forceinline__ __device__ void optix::rt_trace_with_time (
    unsigned int group,
    float3 origin,
```

```
float3 direction,
unsigned int ray_type,
float tmin,
float tmax,
float time,
RTvisibilitymask mask,
RTrayflags flags,
void *prd,
unsigned int prd_size ) [static]
```

6.1.3.574 static __forceinline__ __device__ float3 optix::rt_transform_normal (
RTtransformkind kind,
const float3 & n) [static]

6.1.3.575 static __forceinline__ __device__ float3 optix::rt_transform_point (
RTtransformkind kind,
const float3 & p) [static]

6.1.3.576 static __forceinline__ __device__ float3 optix::rt_transform_vector (
RTtransformkind kind,
const float3 & v) [static]

6.1.3.577 void optix::rt_undefined_use (
int)

6.1.3.578 void optix::rt_undefined_use64 (
int)

6.1.3.579 template<typename T > __device__ T optix::rtTex1D (
rtTextureId id,
float x) [inline]

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, rtTex1DGrad, rtTex2DGrad, rtTex3DGrad, rtTex1DLayeredGrad, rtTex2DLayeredGrad, rtTex1DLod, rtTex2DLod, rtTex3DLod, rtTex1DLayeredLod, rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered.

And cubeamp textures with **rtTexCubemap, rtTexCubemapLod, rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, rtTex2D and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, rtTex1DFetch, rtTex2DFetch, rtTex3DFetch, rtTex2DGather, rtTex1DGrad, rtTex2DGrad, rtTex3DGrad, rtTex1DLayeredGrad, rtTex2DLayeredGrad, rtTex1DLod, rtTex2DLod, rtTex3DLod, rtTex1DLayeredLod, rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered, rtTexCubemap, rtTexCubemapLod, rtTexCubemapLayered and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.580 template<> __device__ float4 optix::rtTex1D (
rtTextureId id,
float x) [inline]

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, rtTex2D and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather, rtTex1DGrad, rtTex2DGrad, rtTex3DGrad, rtTex1DLayeredGrad, rtTex2DLayeredGrad, rtTex1DLod, rtTex2DLod, rtTex3DLod, rtTex1DLayeredLod, rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap, rtTexCubemapLod, rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.581 template<> __device__ int4 optix::rtTex1D (
    rtTextureId id,
    float x ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.582 template<> __device__ uint4 optix::rtTex1D (
    rtTextureId id,
```

```
float x ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.583 **template<typename T > __device__ T optix::rtTex1DFetch (**

```
    rtTextureId id,
    int x ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.584 template<> __device__ float4 optix::rtTex1DFetch (
rtTextureId id,
int x) [inline]

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
```

```
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.585 template<> __device__ int4 optix::rtTex1DFetch (
    rtTextureId id,
    int x ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.586 template<> __device__ uint4 optix::rtTex1DFetch (
    rtTextureId id,
    int x ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.587 template<> __device__ float4 optix::rtTex1DGrad (
    rtTextureId id,
    float x,
    float dPdx,
    float dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There

are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.588 `template<> __device__ int4 optix::rtTex1DGrad (`

```
    rtTextureId id,
    float x,
    float dPdx,
    float dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, rtTex1DGrad, rtTex2DGrad, rtTex3DGrad, rtTex1DLayeredGrad, rtTex2DLayeredGrad, rtTex1DLod, rtTex2DLod, rtTex3DLod, rtTex1DLayeredLod, rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered.

And cubeamp textures with **rtTexCubemap, rtTexCubemapLod, rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, rtTex2D and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, rtTex1DFetch, rtTex2DFetch, rtTex3DFetch, rtTex2DGather, rtTex1DGrad, rtTex2DGrad, rtTex3DGrad, rtTex1DLayeredGrad, rtTex2DLayeredGrad, rtTex1DLod, rtTex2DLod, rtTex3DLod, rtTex1DLayeredLod, rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered, rtTexCubemap, rtTexCubemapLod, rtTexCubemapLayered and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.589 template<> __device__ uint4 optix::rtTex1DGrad (

```
    rtTextureId id,
    float x,
    float dPdx,
    float dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, rtTex2D and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather, rtTex1DGrad, rtTex2DGrad, rtTex3DGrad, rtTex1DLayeredGrad, rtTex2DLayeredGrad, rtTex1DLod, rtTex2DLod, rtTex3DLod, rtTex1DLayeredLod, rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap, rtTexCubemapLod, rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.590 template<typename T> __device__ T optix::rtTex1DLayered (
    rtTextureId id,
    float x,
    int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.591 template<> __device__ float4 optix::rtTex1DLayered (
    rtTextureId id,
    float x,
    int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.592 template<> __device__ int4 optix::rtTex1DLayered (
    rtTextureId id,
    float x,
    int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There

are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.593 template<> __device__ uint4 optix::rtTex1DLayered (
    rtTextureId id,
    float x,
    int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,

rtTex2DLayeredGrad, rtTex1DLod, rtTex2DLod, rtTex3DLod, rtTex1DLayeredLod, rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered.

And cubeamp textures with **rtTexCubemap, rtTexCubemapLod, rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, rtTex2D and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, rtTex1DFetch, rtTex2DFetch, rtTex3DFetch, rtTex2DGather, rtTex1DGrad, rtTex2DGrad, rtTex3DGrad, rtTex1DLayeredGrad, rtTex2DLayeredGrad, rtTex1DLod, rtTex2DLod, rtTex3DLod, rtTex1DLayeredLod, rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered, rtTexCubemap, rtTexCubemapLod, rtTexCubemapLayered and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.594 template<typename T > __device__ T optix::rtTex1DLayeredGrad (

```
    rtTextureId id,
    float x,
    int layer,
    float dPdx,
    float dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, rtTex2D and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.595 template<> __device__ float4 optix::rtTex1DLayeredGrad (
    rtTextureId id,
    float x,
    int layer,
    float dPdx,
    float dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.596 template<> __device__ int4 optix::rtTex1DLayeredGrad (
    rtTextureId id,
    float x,
    int layer,
    float dPdx,
    float dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.597 template<> __device__ uint4 optix::rtTex1DLayeredGrad (
    rtTextureId id,
    float x,
    int layer,
    float dPdx,
```

float *dPdy*) [inline]

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.598 template<typename T > __device__ T optix::rtTex1DLayeredLod (

```
    rtTextureId id,
    float x,
    int layer,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.599 template<> __device__ float4 optix::rtTex1DLayeredLod (
    rtTextureId id,
    float x,
    int layer,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**,

rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.600 template<> __device__ int4 optix::rtTex1DLayeredLod (

```
    rtTextureId id,
    float x,
    int layer,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.601 template<> __device__ uint4 optix::rtTex1DLayeredLod (
    rtTextureId id,
    float x,
    int layer,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.602 template<typename T > __device__ T optix::rtTex1DLod (
    rtTextureId id,
    float x,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.603 template<> __device__ float4 optix::rtTex1DLod (
    rtTextureId id,
    float x,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There

are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.604 template<> __device__ int4 optix::rtTex1DLod (
    rtTextureId id,
    float x,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,

rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and
rtTexCubemapLayeredLod.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**,
rtTex2DGrad, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**,
rtTex2DLod, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**,
rtTex2DLayered, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and
rtTexCubemapLayeredLod were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.605 template<> __device__ uint4 optix::rtTex1DLod (
rtTextureId *id*,
float *x*,
float *level*) [inline]

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.606 template<typename T> __device__ T optix::rtTex2D (
    rtTextureId id,
    float x,
    float y ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.607 template<> __device__ float4 optix::rtTex2D (
```

```
rtTextureId id,
float x,
float y ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.608 template<> __device__ int4 optix::rtTex2D (
    rtTextureId id,
    float x,
    float y ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2,

uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.609 template<> __device__ uint4 optix::rtTex2D (
    rtTextureId id,
    float x,
    float y ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**,

rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.610 template<typename T > __device__ T optix::rtTex2DFetch (
    rtTextureId id,
    int x,
    int y ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.611 template<> __device__ float4 optix::rtTex2DFetch (
    rtTextureId id,
    int x,
    int y ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.612 template<> __device__ int4 optix::rtTex2DFetch (
    rtTextureId id,
```

```
int x,  

int y ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)  
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.613 template<> __device__ uint4 optix::rtTex2DFetch (  
    rtTextureId id,  
    int x,  
    int y ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.614 template<typename T > __device__ T optix::rtTex2DGather (
    rtTextureId id,
    float x,
    float y,
    int comp = 0 ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**,

rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.615 template<> __device__ float4 optix::rtTex2DGather (
    rtTextureId id,
    float x,
    float y,
    int comp ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.616 template<> __device__ int4 optix::rtTex2DGather (
    rtTextureId id,
    float x,
    float y,
    int comp ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.617 template<> __device__ uint4 optix::rtTex2DGather (
    rtTextureId id,
    float x,
    float y,
    int comp ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.618 template<typename T > __device__ T optix::rtTex2DGrad (
    rtTextureId id,
    float x,
    float y,
    float2 dPdx,
    float2 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.619 template<> __device__ float4 optix::rtTex2DGrad (
    rtTextureId id,
    float x,
    float y,
    float2 dPdx,
    float2 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.620 template<> __device__ int4 optix::rtTex2DGrad (
    rtTextureId id,
    float x,
    float y,
    float2 dPdx,
    float2 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *RetVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.621 template<> __device__ uint4 optix::rtTex2DGrad (

```
    rtTextureId id,
    float x,
    float y,
    float2 dPdx,
    float2 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *RetVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.622 template<typename T> __device__ T optix::rtTex2DLayered (
    rtTextureId id,
    float x,
    float y,
    int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.623 template<> __device__ float4 optix::rtTex2DLayered (
    rtTextureId id,
    float x,
    float y,
    int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.624 template<> __device__ int4 optix::rtTex2DLayered (
    rtTextureId id,
    float x,
    float y,
    int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.625 template<> __device__ uint4 optix::rtTex2DLayered (
    rtTextureId id,
    float x,
    float y,
    int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.626 template<typename T > __device__ T optix::rtTex2DLayeredGrad (
    rtTextureId id,
    float x,
    float y,
    int layer,
    float2 dPdx,
    float2 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and

rtTexCubemapLayeredLod.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *RetVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.627 template<> __device__ float4 optix::rtTex2DLayeredGrad (

```
    rtTextureId id,
    float x,
    float y,
    int layer,
    float2 dPdx,
    float2 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *RetVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.628 template<> __device__ int4 optix::rtTex2DLayeredGrad (
    rtTextureId id,
    float x,
    float y,
    int layer,
    float2 dPdx,
    float2 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.629 template<> __device__ uint4 optix::rtTex2DLayeredGrad (
    rtTextureId id,
    float x,
    float y,
    int layer,
    float2 dPdx,
    float2 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.630 template<typename T > __device__ T optix::rtTex2DLayeredLod (
    rtTextureId id,
    float x,
    float y,
```

```
int layer,  

float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)  
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.631 template<> __device__ float4 optix::rtTex2DLayeredLod (
```

<pre>rtTextureId <i>id</i>,</pre> <pre>float <i>x</i>,</pre> <pre>float <i>y</i>,</pre> <pre>int <i>layer</i>,</pre> <pre>float <i>level</i>) [inline]</pre>
--

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There

are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.632 `template<> __device__ int4 optix::rtTex2DLayeredLod (`

```
    rtTextureId id,
    float x,
    float y,
    int layer,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic

filtering. An integer layer number is required for layered textures (arrays of textures) using functions:
rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and
rtTexCubemapLayeredLod.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**,
rtTex2DGrad, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**,
rtTex2DLod, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**,
rtTex2DLayered, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and
rtTexCubemapLayeredLod were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.633 template<> __device__ uint4 optix::rtTex2DLayeredLod (

```
    rtTextureId id,
    float x,
    float y,
    int layer,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:
rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and
rtTexCubemapLayeredLod.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.634 template<typename T > __device__ T optix::rtTex2DLod (
rtTextureId id,
float x,
float y,
float level) [inline]

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**,

rtTex2DLayered, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.635 template<> __device__ float4 optix::rtTex2DLod (
    rtTextureId id,
    float x,
    float y,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.636 template<> __device__ int4 optix::rtTex2DLod (
    rtTextureId id,
    float x,
    float y,
```

float *level*) [inline]

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.637 template<> __device__ uint4 optix::rtTex2DLod (

```
    rtTextureId id,
    float x,
    float y,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.638 template<typename T > __device__ T optix::rtTex3D (
    rtTextureId id,
    float x,
    float y,
    float z ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.639 template<> __device__ float4 optix::rtTex3D (
    rtTextureId id,
    float x,
    float y,
    float z ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.640 template<> __device__ int4 optix::rtTex3D (
    rtTextureId id,
    float x,
    float y,
    float z ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.641 template<> __device__ uint4 optix::rtTex3D (
    rtTextureId id,
    float x,
    float y,
    float z ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.642 template<typename T > __device__ T optix::rtTex3DFetch (
    rtTextureId id,
    int x,
    int y,
    int z ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.643 template<> __device__ float4 optix::rtTex3DFetch (
    rtTextureId id,
    int x,
    int y,
    int z ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**,
rtTex2DGrad, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**,
rtTex2DLod, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**,
rtTex2DLayered, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and
rtTexCubemapLayeredLod were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.644 template<> __device__ int4 optix::rtTex3DFetch (
rtTextureId id,
int x,
int y,
int z) [inline]

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:
rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
```

```
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.645 template<> __device__ uint4 optix::rtTex3DFetch (
    rtTextureId id,
    int x,
    int y,
    int z ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and

rtTexCubemapLayeredLod were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.646 template<typename T > __device__ T optix::rtTex3DGrad (
    rtTextureId id,
    float x,
    float y,
    float z,
    float4 dPdx,
    float4 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.647 template<> __device__ float4 optix::rtTex3DGrad (
    rtTextureId id,
    float x,
```

```
float y,  

float z,  

float4 dPdx,  

float4 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)  
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.648 template<> __device__ int4 optix::rtTex3DGrad (  
    rtTextureId id,  
    float x,  
    float y,  
    float z,  
    float4 dPdx,  
    float4 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.649 template<> __device__ uint4 optix::rtTex3DGrad (
    rtTextureId id,
    float x,
    float y,
    float z,
    float4 dPdx,
    float4 dPdy ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size

reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template< uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.650 template<typename T > __device__ T optix::rtTex3DLod (

```
    rtTextureId id,
    float x,
    float y,
    float z,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**.

rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.651 template<> __device__ float4 optix::rtTex3DLod (

```
    rtTextureId id,
    float x,
    float y,
    float z,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.652 template<> __device__ int4 optix::rtTex3DLod (
    rtTextureId id,
    float x,
    float y,
    float z,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.653 template<> __device__ uint4 optix::rtTex3DLod (
    rtTextureId id,
    float x,
    float y,
    float z,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.654 template<typename T > __device__ T optix::rtTexCubemap (
    rtTextureId id,
    float x,
    float y,
```

```
float z ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.655 **template<> __device__ float4 optix::rtTexCubemap (**

```
    rtTextureId id,
    float x,
    float y,
    float z ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.656 template<> __device__ int4 optix::rtTexCubemap (
    rtTextureId id,
    float x,
    float y,
    float z ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,

rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.657 template<> __device__ uint4 optix::rtTexCubemap (
    rtTextureId id,
    float x,
    float y,
    float z ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.658 template<typename T > __device__ T optix::rtTexCubemapLayered (
    rtTextureId id,
    float x,
    float y,
    float z,
    int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.659 template<> __device__ float4 optix::rtTexCubemapLayered (
    rtTextureId id,
    float x,
    float y,
    float z,
    int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.660 template<> __device__ int4 optix::rtTexCubemapLayered (
    rtTextureId id,
    float x,
    float y,
    float z,
```

```
int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.661 **template<> __device__ uint4 optix::rtTexCubemapLayered (**

```
    rtTextureId id,
    float x,
    float y,
    float z,
    int layer ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2,

uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.662 template<typename T > __device__ T optix::rtTexCubemapLayeredLod (
    rtTextureId id,
    float x,
    float y,
    float z,
    int layer,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**,
rtTex2DGrad, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**,
rtTex2DLod, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**,
rtTex2DLayered, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and
rtTexCubemapLayeredLod were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.663 template<> __device__ float4 optix::rtTexCubemapLayeredLod (
rtTextureId *id*,
float *x*,
float *y*,
float *z*,
int *layer*,
float *level*) [inline]

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:
rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and

rtTexCubemapLayeredLod.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *RetVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.664 template<> __device__ int4 optix::rtTexCubemapLayeredLod (
    rtTextureId id,
    float x,
    float y,
    float z,
    int layer,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *RetVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.665 template<> __device__ uint4 optix::rtTexCubemapLayeredLod (
    rtTextureId id,
    float x,
    float y,
    float z,
    int layer,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.666 template<typename T> __device__ T optix::rtTexCubemapLod (
    rtTextureId id,
    float x,
    float y,
    float z,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.667 template<> __device__ float4 optix::rtTexCubemapLod (
    rtTextureId id,
    float x,
    float y,
    float z,
```

float *level*) [inline]

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.668 template<> __device__ int4 optix::rtTexCubemapLod (

```
    rtTextureId id,
    float x,
    float y,
    float z,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2,

uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

```
6.1.3.669 template<> __device__ uint4 optix::rtTexCubemapLod (
    rtTextureId id,
    float x,
    float y,
    float z,
    float level ) [inline]
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, rtTex1DGrad, rtTex2DGrad, rtTex3DGrad, rtTex1DLayeredGrad, rtTex2DLayeredGrad, rtTex1DLod, rtTex2DLod, rtTex3DLod, rtTex1DLayeredLod, rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered.

And cubeamp textures with **rtTexCubemap, rtTexCubemapLod, rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, rtTex2D and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, rtTex1DFetch, rtTex2DFetch, rtTex3DFetch, rtTex2DGather, rtTex1DGrad, rtTex2DGrad, rtTex3DGrad, rtTex1DLayeredGrad, rtTex2DLayeredGrad, rtTex1DLod, rtTex2DLod, rtTex3DLod, rtTex1DLayeredLod, rtTex2DLayeredLod, rtTex1DLayered, rtTex2DLayered, rtTexCubemap, rtTexCubemapLod, rtTexCubemapLayered and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.3.670 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

```
float1 & v,
int i,
float x )
```

If used on the device, this could place the the 'v' in local memory.

6.1.3.671 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

```
float2 & v,
int i,
float x )
```

If used on the device, this could place the the 'v' in local memory.

6.1.3.672 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

```
float3 & v,
int i,
float x )
```

If used on the device, this could place the the 'v' in local memory.

6.1.3.673 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

```
float4 & v,
int i,
float x )
```

If used on the device, this could place the the 'v' in local memory.

6.1.3.674 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

```
int1 & v,  
int i,  
int x )
```

If used on the device, this could place the the 'v' in local memory.

6.1.3.675 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

```
int2 & v,  
int i,  
int x )
```

If used on the device, this could place the the 'v' in local memory.

6.1.3.676 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

```
int3 & v,  
int i,  
int x )
```

If used on the device, this could place the the 'v' in local memory.

6.1.3.677 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

```
int4 & v,  
int i,  
int x )
```

If used on the device, this could place the the 'v' in local memory.

6.1.3.678 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

```
uint1 & v,  
int i,  
unsigned int x )
```

If used on the device, this could place the the 'v' in local memory.

6.1.3.679 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

```
uint2 & v,  
int i,  
unsigned int x )
```

If used on the device, this could place the the 'v' in local memory.

6.1.3.680 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

```
uint3 & v,  
int i,  
unsigned int x )
```

If used on the device, this could place the the 'v' in local memory.

6.1.3.681 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

uint4 & v,

int i,

unsigned int x)

If used on the device, this could place the the 'v' in local memory.

6.1.3.682 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

longlong1 & v,

int i,

long long x)

If used on the device, this could place the the 'v' in local memory.

6.1.3.683 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

longlong2 & v,

int i,

long long x)

If used on the device, this could place the the 'v' in local memory.

6.1.3.684 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

longlong3 & v,

int i,

int x)

If used on the device, this could place the the 'v' in local memory.

6.1.3.685 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

longlong4 & v,

int i,

long long x)

If used on the device, this could place the the 'v' in local memory.

6.1.3.686 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

ulonglong1 & v,

int i,

unsigned long long x)

If used on the device, this could place the the 'v' in local memory.

6.1.3.687 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (

ulonglong2 & v,

int i,

```
unsigned long long x )
```

If used on the device, this could place the the 'v' in local memory.

```
6.1.3.688 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (
    ulonglong3 & v,
    int i,
    unsigned long long x )
```

If used on the device, this could place the the 'v' in local memory.

```
6.1.3.689 OPTIXU_INLINE RT_HOSTDEVICE void optix::setByIndex (
    ulonglong4 & v,
    int i,
    unsigned long long x )
```

If used on the device, this could place the the 'v' in local memory.

```
6.1.3.690 OPTIXU_INLINE RT_HOSTDEVICE float optix::smoothstep (
    const float edge0,
    const float edge1,
    const float x )
```

Return a smooth value in [0,1], where the transition from 0 to 1 takes place for values of x in [edge0,edge1].

```
6.1.3.691 OPTIXU_INLINE RT_HOSTDEVICE float2 optix::square_to_disk (
    const float2 & sample )
```

Maps concentric squares to concentric circles (Shirley and Chiu)

```
6.1.3.692 OPTIXU_INLINE RT_HOSTDEVICE float3 optix::temperature (
    const float t )
```

Simple mapping from [0,1] to a temperature-like RGB color.

6.1.4 Variable Documentation

6.1.4.1 rtTextureId float float int optix::comp

Initial value:

```
= 0)
{
    T tmp = rtTex2DGather<T>(id, x, y, comp);
    *retVal = tmp;
}
```

```
template<typename T>
inline __device__ T rtTex1DGrad(rtTextureId id, float x, float
dPdx, float dPdy)
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.4.2 rtTextureId float float int float2 optix::dPdx

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size

reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.4.3 rtTextureId float float int float2 float2 optix::dPdy

Initial value:

```
{
    T tmp = rtTex1DGrad<T>(id, x, dPdx, dPdy)
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.4.4 rtTextureId optix::id

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.4.5 const unsigned char optix::KEY[7] = {164, 195, 147, 255, 203, 161, 184}

6.1.4.6 rtTextureId float float float int optix::layer

Initial value:

```
{  
    T tmp = rtTex1DLayered<T>(id, x, layer)
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)  
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.4.7 rtTextureId float float float int float optix::level

Initial value:

```
{
```

```
T tmp = rtTex1DLod<T>(id, x, level)
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.4.8 * optix::RetVal = tmp

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using **rtTextureSamplerGetId** function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:

rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.4.9 rtTextureId float optix::x

Initial value:

```
{  
    T tmp = rtTex1D<T>(id, x)
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions:
rtTex2DGather, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**,
rtTex2DLayeredGrad, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**,
rtTex2DLayeredLod, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.4.10 rtTextureId float float optix::y

Initial value:

```
{  
    T tmp = rtTex2D<T>(id, x, y)
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**,

rtTex2DLod, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.1.4.11 **rtTextureId float float float optix::z**

Initial value:

```
{
    T tmp = rtTex3D<T>(id, x, y, z)
```

Similar to CUDA C's texture functions, OptiX programs can access textures in a bindless way.

Description

rtTex1D, **rtTex2D** and **rtTex3D** fetch the texture referenced by the *id* with texture coordinate *x*, *y* and *z*. The texture sampler *id* can be obtained on the host side using [rtTextureSamplerGetId](#) function. There are also C++ template and C-style additional declarations for other texture types (char1, uchar1, char2, uchar2 ...):

To get texture size dimensions **rtTexSize** can be used. In the case of compressed textures, the size reflects the full view size, rather than the compressed data size.

Texture element may be fetched with integer coordinates using functions: **rtTex1DFetch**, **rtTex2DFetch** and **rtTex3DFetch**

Textures may also be sampled by providing a level of detail for mip mapping or gradients for anisotropic filtering. An integer layer number is required for layered textures (arrays of textures) using functions: **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**.

And cubeamp textures with **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod**.

```
template<> uchar2 rtTex1D(rtTextureId id, float x)
void rtTex1D(ushort2 *retVal, rtTextureId id, float x)
```

History

rtTex1D, **rtTex2D** and **rtTex3D** were introduced in OptiX 3.0.

rtTexSize, **rtTex1DFetch**, **rtTex2DFetch**, **rtTex3DFetch**, **rtTex2DGather**, **rtTex1DGrad**, **rtTex2DGrad**, **rtTex3DGrad**, **rtTex1DLayeredGrad**, **rtTex2DLayeredGrad**, **rtTex1DLod**, **rtTex2DLod**, **rtTex3DLod**, **rtTex1DLayeredLod**, **rtTex2DLayeredLod**, **rtTex1DLayered**, **rtTex2DLayered**, **rtTexCubemap**, **rtTexCubemapLod**, **rtTexCubemapLayered** and **rtTexCubemapLayeredLod** were introduced in OptiX 3.9.

See also [rtTextureSamplerGetId](#)

6.2 optix::detail Namespace Reference

Functions

- void [generateSalt](#) (unsigned char **buffer*)
- void [sha256](#) (const unsigned char **input*, unsigned int *input_length*, unsigned char **buffer*)

6.2.1 Function Documentation

6.2.1.1 void optix::detail::generateSalt (
 unsigned char * *buffer*) [inline]

6.2.1.2 void optix::detail::sha256 (
 const unsigned char * *input*,
 unsigned int *input_length*,
 unsigned char * *buffer*) [inline]

6.3 optix::prime Namespace Reference

Classes

- class [ContextObj](#)
- class [BufferDescObj](#)
- class [ModelObj](#)
- class [QueryObj](#)
- class [Exception](#)

Typedefs

- [typedef Handle< BufferDescObj > BufferDesc](#)
- [typedef Handle< ContextObj > Context](#)
- [typedef Handle< ModelObj > Model](#)
- [typedef Handle< QueryObj > Query](#)

Functions

- [std::string getVersionString \(\)](#)
- [void checkError \(RTPresult code\)](#)
- [void checkError \(RTPresult code, RTPcontext context\)](#)

6.3.1 Function Documentation

6.3.1.1 void optix::prime::checkError (

RTPresult *code*) [inline]

6.3.1.2 void `optix::prime::checkError(RTPresult code,`
`RTPcontext context) [inline]`

6.3.1.3 std::string `optix::prime::getVersionString() [inline]`

Returns a string describing the version of the OptiX Prime being used. See [rtpGetVersionString](#).

6.4 optixu Namespace Reference

6.5 rti_internal_callableprogram Namespace Reference

Classes

- class `CPArgVoid`
- struct `is_CPAvgVoid`
- struct `is_CPAvgVoid< CPArgVoid >`
- struct `check_is_CPAvgVoid`
- struct `check_is_CPAvgVoid< false, IntentionalError >`
- class `callableProgramIdBase`
- class `markedCallableProgramIdBase`

6.6 rti_internal_typeinfo Namespace Reference

Classes

- struct `rti_typeinfo`
- struct `rti_typeenum`
- struct `rti_typeenum< optix::callableProgramId< T > >`
- struct `rti_typeenum< optix::boundCallableProgramId< T > >`

Enumerations

- enum `rtiTypeKind` { `_OPTIX_VARIABLE` = 0x796152 }
- enum `rtiTypeEnum` {
 `_OPTIX_TYPE_ENUM_UNKNOWN` = 0x1337,
 `_OPTIX_TYPE_ENUM_PROGRAM_ID`,
 `_OPTIX_TYPE_ENUM_PROGRAM_AS_ID` }

6.6.1 Enumeration Type Documentation

6.6.1.1 enum rti_internal_typeinfo::rtiTypeEnum

Enumerator

```
_OPTIX_TYPE_ENUM_UNKNOWN
_OPTIX_TYPE_ENUM_PROGRAM_ID
_OPTIX_TYPE_ENUM_PROGRAM_AS_ID
```

6.6.1.2 enum rti_internal_typeinfo::rtiTypeKind

Enumerator

```
_OPTIX_VARIABLE
```

7 Class Documentation

7.1 optix::Aabb Class Reference

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

7.1.1 Detailed Description

Axis-aligned bounding box.

Description

Aabb is a utility class for computing and manipulating axis-aligned bounding boxes (aabbs). 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](#)

7.1.2 Constructor & Destructor Documentation

7.1.2.1 OPTIXU_INLINE RT_HOSTDEVICE optix::Aabb::Aabb()

Construct an invalid box.

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

Construct from min and max vectors.

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

Construct from three points (e.g.

triangle)

7.1.3 Member Function Documentation

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

Compute the surface area of the box.

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

Compute the box center.

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

Compute the box center in the given dimension.

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

Check if the point is in the box.

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

Check if the box is fully contained in the box.

7.1.3.6 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.

7.1.3.7 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.

7.1.3.8 OPTIXU_INLINE RT_HOSTDEVICE void optix::Aabb::enlarge(float amount)

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

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

Compute the box extent.

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

Compute the box extent in the given dimension.

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

Compute half the surface area of the box.

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

Extend the box to include the given point.

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

Extend the box to include the given box.

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

Extend the box to include the given box.

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

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

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

Check for intersection with another box.

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

Invalidate the box.

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

Check if the box is flat in at least one dimension.

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

Get the index of the longest axis.

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

Get the extent of the longest axis.

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

Exact equality.

7.1.3.22]

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

Array access.

7.1.3.23]

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

Const array access.

```
7.1.3.24 OPTIXU_INLINE RT_HOSTDEVICE void optix::Aabb::set ( const float3 & min, const float3 & max )
```

Set using two vectors.

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

Set using three points (e.g.
triangle)

```
7.1.3.26 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

```
7.1.3.27 OPTIXU_INLINE RT_HOSTDEVICE bool optix::Aabb::valid ( ) const
```

Check if the box is valid.

```
7.1.3.28 OPTIXU_INLINE RT_HOSTDEVICE float optix::Aabb::volume ( ) const
```

Compute the volume of the box.

7.1.4 Member Data Documentation

```
7.1.4.1 float3 optix::Aabb::m_max
```

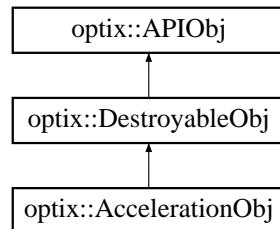
Max bound.

7.1.4.2 float3 optix::Aabb::m_min

Min bound.

7.2 optix::AccelerationObj Class Reference

Inheritance diagram for optix::AccelerationObj:



Public Member Functions

- void `destroy ()`
- void `validate ()`
- `Context getContext () const`
- `RTacceleration get ()`
- void `addReference ()`
- int `removeReference ()`
- virtual void `checkError (RTresult code) const`
- virtual void `checkError (RTresult code, Context context) const`
- void `checkErrorNoGetContext (RTresult code) const`

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

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

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

Static Public Member Functions

- static `Exception makeException (RTresult code, RTcontext context)`

7.2.1 Detailed Description

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

7.2.2 Member Function Documentation

7.2.2.1 void optix::APIObj::addReference() [inline], [inherited]

Increment the reference count for this object.

7.2.2.2 void optix::APIObj::checkError(RTresult code) const [inline], [virtual], [inherited]

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

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

7.2.2.3 void optix::APIObj::checkError(RTresult code, Context context) const [inline], [virtual], [inherited]

7.2.2.4 void optix::APIObj::checkErrorNoGetContext(RTresult code) const [inline], [inherited]

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

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

Implements [optix::DestroyableObj](#).

7.2.2.6 RTacceleration optix::AccelerationObj::get() [inline]

Get the underlying OptiX C API RTacceleration opaque pointer.

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

Query the acceleration structure builder. See [rtAccelerationGetBuilder](#).

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

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).

Implements [optix::APIObj](#).

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

Deprecated in OptiX 4.0 Get the marshaled acceleration data. See [rtAccelerationGetData](#).

7.2.2.10 RTsize optix::AccelerationObj::getDataSetSize() const [inline]

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

7.2.2.11 std::string optix::AccelerationObj::getProperty(

const std::string & name) const [inline]

Query properties specifying Acceleration builder behavior.

See [rtAccelerationGetProperty](#).

7.2.2.12 std::string optix::AccelerationObj::getTraverser() const [inline]

Deprecated in OptiX 4.0 Query the acceleration structure traverser. See [rtAccelerationGetTraverser](#).

7.2.2.13 bool optix::AccelerationObj::isDirty() const [inline]

Query if the acceleration needs a rebuild. See [rtAccelerationIsDirty](#).

7.2.2.14 Exception optix::APIObj::makeException(

RTresult code,

RTcontext context) [inline], [static], [inherited]

For backwards compatibility. Use [Exception::makeException](#) instead.

7.2.2.15 void optix::AccelerationObj::markDirty() [inline]

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

7.2.2.16 int optix::APIObj::removeReference() [inline], [inherited]

Decrement the reference count for this object.

7.2.2.17 void optix::AccelerationObj::setBuilder(

const std::string & builder) [inline]

Specify the acceleration structure builder. See [rtAccelerationSetBuilder](#).

7.2.2.18 void optix::AccelerationObj::setData(

const void * data,

RTsize size) [inline]

Deprecated in OptiX 4.0 Specify the acceleration structure via marshaled acceleration data. See [rtAccelerationSetData](#).

7.2.2.19 void optix::AccelerationObj::setProperty(

const std::string & name,

const std::string & value) [inline]

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

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

Deprecated in OptiX 4.0 Specify the acceleration structure traverser. See [rtAccelerationSetTraverser](#).

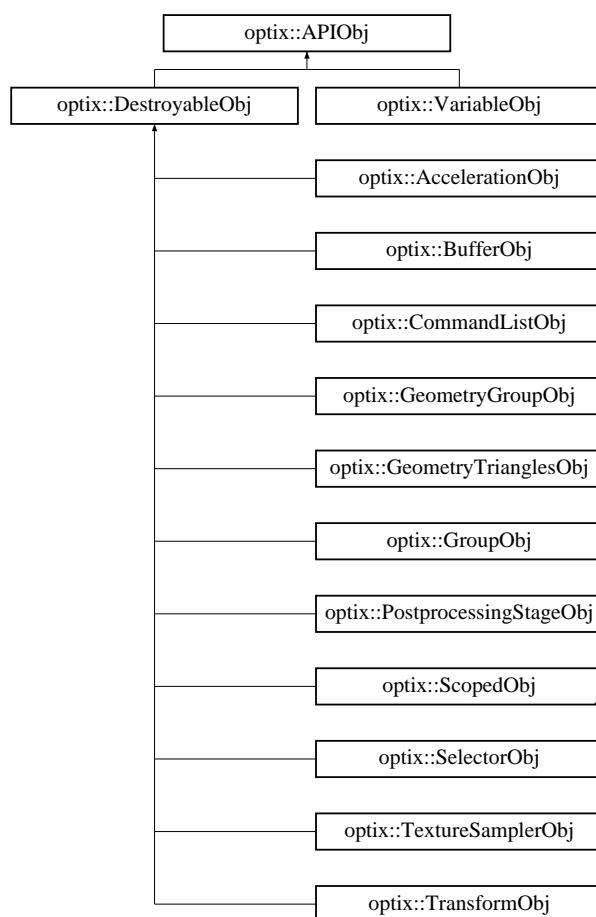
7.2.2.21 void optix::AccelerationObj::validate() [inline], [virtual]

call [rt\[ObjectType\]Validate](#) on the underlying OptiX C object

Implements [optix::DestroyableObj](#).

7.3 optix::APIObj Class Reference

Inheritance diagram for [optix::APIObj](#):



Public Member Functions

- [APIObj\(\)](#)
- [virtual ~APIObj\(\)](#)
- [void addReference\(\)](#)
- [int removeReference\(\)](#)
- [virtual Context getContext\(\) const =0](#)
- [virtual void checkError\(RTresult code\) const](#)

- virtual void `checkError (RTresult code, Context context) const`
- void `checkErrorNoGetContext (RTresult code) const`

Static Public Member Functions

- static `Exception makeException (RTresult code, RTcontext context)`

7.3.1 Detailed Description

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

Wraps:

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

7.3.2 Constructor & Destructor Documentation

7.3.2.1 `optix::APIObj::APIObj() [inline]`

7.3.2.2 `virtual optix::APIObj::~APIObj() [inline], [virtual]`

7.3.3 Member Function Documentation

7.3.3.1 `void optix::APIObj::addReference() [inline]`

Increment the reference count for this object.

7.3.3.2 `void optix::APIObj::checkError (RTResult code) const [inline], [virtual]`

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

Reimplemented in `optix::ContextObj`.

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

7.3.3.4 void optix::APIObj::checkErrorNoGetContext (
RTresult code) const [inline]

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

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

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

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

For backwards compatibility. Use `Exception::makeException` instead.

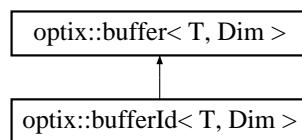
7.3.3.7 int optix::APIObj::removeReference() [inline]

Decrement the reference count for this object.

7.4 optix::boundCallableProgramId< T > Class Template Reference

7.5 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)`

7.5.1 Member Typedef Documentation

7.5.1.1 template<typename T, int Dim = 1> typedef VectorTypes<size_t, Dim>::Type optix::buffer< T, Dim >::IndexType

7.5.1.2 template<typename T, int Dim = 1> typedef VectorTypes<size_t, Dim> optix::buffer< T, Dim >::WrapperType

7.5.2 Member Function Documentation

7.5.2.1 template<typename T, int Dim = 1> template<typename T2 > __device__ static __forceinline__ void* optix::buffer< T, Dim >::create (type< T2 > , void * v) [inline], [static], [protected]

7.5.2.2 template<typename T, int Dim = 1> template<typename T2 , int Dim2> __device__ static __forceinline__ void* optix::buffer< T, Dim >::create (type< bufferId< T2, Dim2 > > , void * v) [inline], [static], [protected]

7.5.2.3 template<typename T, int Dim = 1> __inline__ static __device__ size_t4 optix::buffer< T, Dim >::make_index (

```
size_t v0 ) [inline], [static], [protected]
```

7.5.2.4 template<typename T, int Dim = 1> __inline__ static __device__ size_t4 optix::buffer< T, Dim >::make_index (
size_t2 v0) [inline], [static], [protected]

7.5.2.5 template<typename T, int Dim = 1> __inline__ static __device__ size_t4 optix::buffer< T, Dim >::make_index (
size_t3 v0) [inline], [static], [protected]

7.5.2.6 template<typename T, int Dim = 1> __inline__ static __device__ size_t4 optix::buffer< T, Dim >::make_index (
size_t4 v0) [inline], [static], [protected]

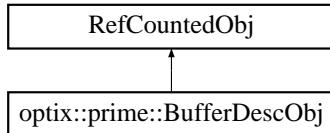
7.5.2.7]

template<typename T, int Dim = 1> __device__ __forceinline__ T& optix::buffer< T, Dim >::operator[] (
IndexType i) [inline]

7.5.2.8 template<typename T, int Dim = 1> __device__ __forceinline__ IndexType
optix::buffer< T, Dim >::size() const [inline]

7.6 optix::prime::BufferDescObj Class Reference

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 \(\)](#)

7.6.1 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.

7.6.2 Member Function Documentation

7.6.2.1 Context optix::prime::BufferDescObj::getContext() [inline]

Returns the context associated within this object.

7.6.2.2 RTPbufferdesc optix::prime::BufferDescObj::getRTPbufferdesc() [inline]

Returns the RTPbufferdesc descriptor stored within this object.

7.6.2.3 void optix::prime::BufferDescObj::setCudaDeviceNumber(unsigned deviceNumber) [inline]

Sets the CUDA device number for a buffer. See [rtpBufferDescSetCudaDeviceNumber](#).

7.6.2.4 void optix::prime::BufferDescObj::setRange(RTPsize begin, RTPsize end) [inline]

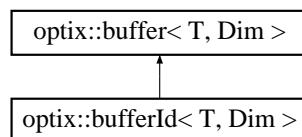
Sets the range of a buffer to be used. See [rtpBufferDescSetRange](#).

7.6.2.5 void optix::prime::BufferDescObj::setStride(unsigned strideBytes) [inline]

Sets the stride for elements in a buffer. See [rtpBufferDescSetStride](#).

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

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()`
- `__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 ()`
- `bufferId (int id)`
- `int getId () 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)`

7.7.1 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;
}
```

7.7.2 Member Typedef Documentation

7.7.2.1 template<typename T, int Dim> typedef buffer<T,Dim>::IndexType optix::bufferId< T, Dim >::IndexType

7.7.2.2 template<typename T, int Dim> typedef buffer<T,Dim>::WrapperType optix::bufferId< T, Dim >::WrapperType

7.7.3 Constructor & Destructor Documentation

7.7.3.1 template<typename T, int Dim> __device__ __forceinline__ optix::bufferId< T, Dim >::bufferId() [inline]

7.7.3.2 template<typename T, int Dim> __device__ __forceinline__ optix::bufferId< T, Dim >::bufferId(RTbufferidnull nullid) [inline]

7.7.3.3 template<typename T, int Dim> __device__ __forceinline__ optix::bufferId< T, Dim >::bufferId(int id) [inline], [explicit]

7.7.3.4 template<typename T, int Dim> optix::bufferId< T, Dim >::bufferId() [inline]

7.7.3.5 template<typename T, int Dim> optix::bufferId< T, Dim >::bufferId(

```
int id ) [inline]
```

7.7.4 Member Function Documentation

7.7.4.1 `template<typename T, int Dim = 1> template<typename T2> __device__ static __forceinline__ void* optix::buffer< T, Dim >::create (type< T2 > , void * v) [inline], [static], [protected], [inherited]`

7.7.4.2 `template<typename T, int Dim = 1> template<typename T2, int Dim2> __device__ static __forceinline__ void* optix::buffer< T, Dim >::create (type< bufferId< T2, Dim2 > > , void * v) [inline], [static], [protected], [inherited]`

7.7.4.3 `template<typename T, int Dim> __device__ __forceinline__ int optix::bufferId< T, Dim >::getId () const [inline]`

7.7.4.4 `template<typename T, int Dim> int optix::bufferId< T, Dim >::getId () const [inline]`

7.7.4.5 `template<typename T, int Dim = 1> __inline__ static __device__ size_t4 optix::buffer< T, Dim >::make_index (size_t v0) [inline], [static], [protected], [inherited]`

7.7.4.6 `template<typename T, int Dim = 1> __inline__ static __device__ size_t4 optix::buffer< T, Dim >::make_index (size_t2 v0) [inline], [static], [protected], [inherited]`

7.7.4.7 `template<typename T, int Dim = 1> __inline__ static __device__ size_t4 optix::buffer< T, Dim >::make_index (size_t3 v0) [inline], [static], [protected], [inherited]`

7.7.4.8 `template<typename T, int Dim = 1> __inline__ static __device__ size_t4 optix::buffer< T, Dim >::make_index (size_t4 v0) [inline], [static], [protected], [inherited]`

7.7.4.9 `template<typename T, int Dim> __device__ __forceinline__ optix::bufferId< T, Dim >::operator bool () const [inline]`

7.7.4.10 `template<typename T, int Dim> __device__ __forceinline__ bufferId& optix::bufferId< T, Dim >::operator= (RTbufferIdnull nullid) [inline]`

7.7.4.11]

`template<typename T, int Dim = 1> __device__ __forceinline__ T& optix::buffer< T, Dim >::operator[] (IndexType i) [inline], [inherited]`

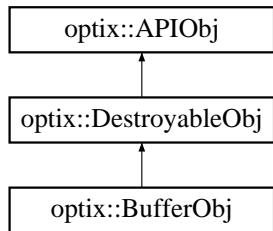
7.7.4.12]

```
template<typename T, int Dim> __device__ __forceinline__ T& optix::bufferId< T, Dim >::operator[] ( IndexType i ) const [inline]
```

7.7.4.13 template<typename T, int Dim> __device__ __forceinline__ **IndexType** optix::bufferId< T, Dim >::size () const [inline]

7.8 optix::BufferObj Class Reference

Inheritance diagram for optix::BufferObj:



Public Member Functions

- void [destroy \(\)](#)
- void [validate \(\)](#)
- Context [getContext \(\) const](#)
- RTbuffer [get \(\)](#)
- void [addReference \(\)](#)
- int [removeReference \(\)](#)
- virtual void [checkError \(RTresult code\) const](#)
- virtual void [checkError \(RTresult code, Context context\) const](#)
- void [checkErrorNoGetContext \(RTresult code\) const](#)

- void [setFormat \(RTformat format\)](#)
- RTformat [getFormat \(\) const](#)
- void [setSize \(RTsize size_of_element\)](#)
- 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, const 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)`

Static Public Member Functions

- static `Exception makeException (RTresult code, RTcontext context)`

7.8.1 Detailed Description

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

7.8.2 Member Function Documentation

7.8.2.1 void optix::APIObj::addReference () [inline], [inherited]

Increment the reference count for this object.

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

Bind a buffer as source for a progressive stream. See `rtBufferBindProgressiveStream`.

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

Check the given result code and throw an error with appropriate message if the code is not RTsuccess.
Reimplemented in [optix::ContextObj](#).

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

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

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

call `rt[ObjectType]Destroy` on the underlying OptiX C object
Implements [optix::DestroyableObj](#).

7.8.2.7 RTbuffer optix::BufferObj::get() [inline]

Get the underlying OptiX C API RTbuffer opaque pointer.

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

Get a Buffer Attribute. See [rtBufferGetAttribute](#).

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

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).
Implements [optix::APIObj](#).

7.8.2.10 void optix::BufferObj::getDevicePointer (
 int *optix_device_ordinal*,
 void ** *device_pointer*) [inline]

Get the pointer to buffer memory on a specific device. See [rtBufferGetDevicePointer](#).

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

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

7.8.2.12 unsigned int optix::BufferObj::getDimensionality() const [inline]

Query dimensionality of buffer. See [rtBufferGetDimensionality](#).

7.8.2.13 RTsize optix::BufferObj::getElementSize() const [inline]

Query the data element size for user format buffers. See [rtBufferGetElementSize](#).

7.8.2.14 RTformat optix::BufferObj::getFormat() const [inline]

Query the data format for the buffer. See [rtBufferGetFormat](#).

7.8.2.15 unsigned int optix::BufferObj::getGLBOld() const [inline]

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

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

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

7.8.2.17 unsigned int optix::BufferObj::getMipLevelCount() const [inline]

Query number of mipmap levels of buffer. See [rtBufferGetMipLevelCount](#).

7.8.2.18 void optix::BufferObj::getMipLevelSize(
unsigned int *level*,
RTsize & *width*) const [inline]

Query 1D buffer dimension of specific MIP level. See [rtBufferGetMipLevelSize1D](#).

7.8.2.19 void optix::BufferObj::getMipLevelSize(
unsigned int *level*,
RTsize & *width*,
RTsize & *height*) const [inline]

Query 2D buffer dimension of specific MIP level. See [rtBufferGetMipLevelSize2D](#).

7.8.2.20 void optix::BufferObj::getMipLevelSize(
unsigned int *level*,
RTsize & *width*,
RTsize & *height*,
RTsize & *depth*) const [inline]

Query 3D buffer dimension of specific MIP level. See [rtBufferGetMipLevelSize3D](#).

7.8.2.21 void optix::BufferObj::getProgressiveUpdateReady(
int * *ready*,
unsigned int * *subframe_count*,
unsigned int * *max_subframes*) [inline]

Query updates from a progressive stream. See [rtBufferGetProgressiveUpdateReady](#).

7.8.2.22 bool optix::BufferObj::getProgressiveUpdateReady() [inline]

Query updates from a progressive stream. See [rtBufferGetProgressiveUpdateReady](#).

7.8.2.23 bool optix::BufferObj::getProgressiveUpdateReady(unsigned int & *subframe_count*) [inline]

Query updates from a progressive stream. See [rtBufferGetProgressiveUpdateReady](#).

7.8.2.24 bool optix::BufferObj::getProgressiveUpdateReady(unsigned int & *subframe_count*, unsigned int & *max_subframes*) [inline]

Query updates from a progressive stream. See [rtBufferGetProgressiveUpdateReady](#).

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

Query 1D buffer dimension. See [rtBufferGetSize1D](#).

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

Query 2D buffer dimension. See [rtBufferGetSize2D](#).

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

Query 3D buffer dimension. See [rtBufferGetSize3D](#).

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

Query dimensions of buffer. See [rtBufferGetSizev](#).

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

For backwards compatibility. Use [Exception::makeException](#) instead.

7.8.2.30 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](#).

7.8.2.31 void optix::BufferObj::markDirty() [inline]

Mark the buffer dirty.

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

Declare the buffer as mutable and inaccessible by OptiX. See [rtTextureSamplerGLRegister](#).

7.8.2.33 int optix::APIObj::removeReference() [inline], [inherited]

Decrement the reference count for this object.

7.8.2.34 void optix::BufferObj::setAttribute(

```
RTbufferattribute attrib,  
RTsize size,  
const void * p ) [inline]
```

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

7.8.2.35 void optix::BufferObj::setDevicePointer(

```
int optix_device_ordinal,  
void * device_pointer ) [inline]
```

Set the pointer to buffer memory on a specific device. See [rtBufferSetDevicePointer](#).

7.8.2.36 void optix::BufferObj::setElementSize(

```
RTsize size_of_element ) [inline]
```

Set the data element size for user format buffers. See [rtBufferSetElementSize](#).

7.8.2.37 void optix::BufferObj::setFormat(

```
RTformat format ) [inline]
```

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

7.8.2.38 void optix::BufferObj::setMipLevelCount(

```
unsigned int levels ) [inline]
```

Set buffer number of MIP levels. See [rtBufferSetMipLevelCount](#).

7.8.2.39 void optix::BufferObj::setSize(

```
RTsize width ) [inline]
```

Set buffer dimensionality to one and buffer width to specified width. See [rtBufferSetSize1D](#).

7.8.2.40 void optix::BufferObj::setSize(

RTsize width,
RTsize height) [inline]

Set buffer dimensionality to two and buffer dimensions to specified width,height. See [rtBufferSetSize2D](#).

7.8.2.41 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](#).

7.8.2.42 void optix::BufferObj::setSize (

unsigned int dimensionality,
const RTsize * dims) [inline]

Set buffer dimensionality and dimensions to specified values. See [rtBufferSetSizev](#).

7.8.2.43 void optix::BufferObj::unmap (

unsigned int level = 0) [inline]

Unmaps a buffer object. See [rtBufferUnmap](#) and [rtBufferUnmapEx](#).

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

Unregister the buffer, re-enabling OptiX operations. See [rtTextureSamplerGLUnregister](#).

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

call [rt\[ObjectType\]Validate](#) on the underlying OptiX C object

Implements [optix::DestroyableObj](#).

7.9 optix::callableProgramId< T > Class Template Reference

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

Public Member Functions

- `__device__ __forceinline__ callableProgramIdBase ()`
- `__device__ __forceinline__ callableProgramIdBase (RTprogramidnull nullid)`
- `__device__ __forceinline__ callableProgramIdBase (int id)`
- `__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`

7.10.1 Constructor & Destructor Documentation

7.10.1.1 `template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::callableProgramIdBase ()
[inline]`

7.10.1.2 `template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::callableProgramIdBase ()`

RTprogramidnull *nullid*) [inline]

7.10.1.3 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::callableProgramIdBase (

int *id*) [inline], [explicit]

7.10.2 Member Function Documentation

7.10.2.1 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() () [inline]

7.10.2.2 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator()

Arg0T arg0) [inline]

7.10.2.3 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (

Arg0T arg0,

Arg1T arg1) [inline]

7.10.2.4 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (

Arg0T arg0,

Arg1T arg1,

Arg2T arg2) [inline]

7.10.2.5 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (

Arg0T arg0,
Arg1T arg1,
Arg2T arg2,
Arg3T arg3) [inline]

7.10.2.6 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (

Arg0T arg0,
Arg1T arg1,
Arg2T arg2,
Arg3T arg3,

Arg4T arg4) [inline]

7.10.2.7 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (

Arg0T arg0,
Arg1T arg1,
Arg2T arg2,
Arg3T arg3,
Arg4T arg4,
Arg5T arg5) [inline]

7.10.2.8 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (

Arg0T arg0,
Arg1T arg1,
Arg2T arg2,
Arg3T arg3,
Arg4T arg4,
Arg5T arg5,

Arg6T arg6) [inline]

7.10.2.9 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (

Arg0T arg0,
Arg1T arg1,
Arg2T arg2,
Arg3T arg3,
Arg4T arg4,
Arg5T arg5,
Arg6T arg6,
Arg7T arg7) [inline]

7.10.2.10 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (

Arg0T arg0,
Arg1T arg1,
Arg2T arg2,
Arg3T arg3,
Arg4T arg4,

```
Arg5T arg5,  
Arg6T arg6,  
Arg7T arg7,  
Arg8T arg8 ) [inline]
```

```
7.10.2.11 template<typename ReturnT , typename Arg0T =  
rti_internal_callableprogram::CPArgVoid, typename Arg1T =  
rti_internal_callableprogram::CPArgVoid, typename Arg2T =  
rti_internal_callableprogram::CPArgVoid, typename Arg3T =  
rti_internal_callableprogram::CPArgVoid, typename Arg4T =  
rti_internal_callableprogram::CPArgVoid, typename Arg5T =  
rti_internal_callableprogram::CPArgVoid, typename Arg6T =  
rti_internal_callableprogram::CPArgVoid, typename Arg7T =  
rti_internal_callableprogram::CPArgVoid, typename Arg8T =  
rti_internal_callableprogram::CPArgVoid, typename Arg9T =  
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT  
rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T,  
Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (  
    Arg0T arg0,  
    Arg1T arg1,  
    Arg2T arg2,  
    Arg3T arg3,  
    Arg4T arg4,  
    Arg5T arg5,  
    Arg6T arg6,  
    Arg7T arg7,  
    Arg8T arg8,
```

Arg9T arg9) [inline]

7.10.3 Member Data Documentation

7.10.3.1 **template<typename ReturnT , typename Arg0T = rti_internal_callableprogram::CPArgVoid, typename Arg1T = rti_internal_callableprogram::CPArgVoid, typename Arg2T = rti_internal_callableprogram::CPArgVoid, typename Arg3T = rti_internal_callableprogram::CPArgVoid, typename Arg4T = rti_internal_callableprogram::CPArgVoid, typename Arg5T = rti_internal_callableprogram::CPArgVoid, typename Arg6T = rti_internal_callableprogram::CPArgVoid, typename Arg7T = rti_internal_callableprogram::CPArgVoid, typename Arg8T = rti_internal_callableprogram::CPArgVoid, typename Arg9T = rti_internal_callableprogram::CPArgVoid> int rti_internal_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::m_id [protected]**

7.11 rti_internal_callableprogram::check_is_CPAvgVoid< Condition, Dummy > Struct Template Reference

Public Types

- **typedef bool result**

7.11.1 Member Typedef Documentation

7.11.1.1 **template<bool Condition, typename Dummy = void> typedef bool rti_internal_callableprogram::check_is_CPAvgVoid< Condition, Dummy >::result**

7.12 rti_internal_callableprogram::check_is_CPAvgVoid< false, IntentionalError > Struct Template Reference

Public Types

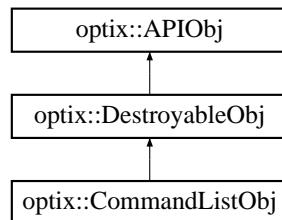
- **typedef IntentionalError::does_not_exist result**

7.12.1 Member Typedef Documentation

7.12.1.1 `template<typename IntentionalError > typedef IntentionalError::does_not_exist rti_internal_callableprogram::check_is_CPAVoid< false, IntentionalError >::result`

7.13 optix::CommandListObj Class Reference

Inheritance diagram for optix::CommandListObj:



Public Member Functions

- `void destroy ()`
- `void validate ()`
- `Context getContext () const`
- `RTcommandlist get ()`
- `void addReference ()`
- `int removeReference ()`
- `virtual void checkError (RTresult code) const`
- `virtual void checkError (RTresult code, Context context) const`
- `void checkErrorNoGetContext (RTresult code) const`

- `void appendPostprocessingStage (PostprocessingStage stage, RTsize launch_width, RTsize launch_height)`
- `void appendLaunch (unsigned int entryIndex, RTsize launch_width, RTsize launch_height)`

- `void finalize ()`
- `void execute ()`

Static Public Member Functions

- `static Exception makeException (RTresult code, RTcontext context)`

7.13.1 Detailed Description

CommandList wraps the OptiX C API RTcommandlist opaque type and its associated function set.

7.13.2 Member Function Documentation

7.13.2.1 void optix::APIObj::addReference() [inline], [inherited]

Increment the reference count for this object.

7.13.2.2 void optix::CommandListObj::appendLaunch(

```
    unsigned int entryIndex,  
    RTsize launch_width,  
    RTsize launch_height ) [inline]
```

Append a launch2d command to the command list. See [rtCommandListAppendLaunch2D](#).

7.13.2.3 void optix::CommandListObj::appendPostprocessingStage(

```
    PostprocessingStage stage,  
    RTsize launch_width,  
    RTsize launch_height ) [inline]
```

Append a postprocessing stage to the command list. See [rtCommandListAppendPostprocessingStage](#).

7.13.2.4 void optix::APIObj::checkError(

```
    RTResult code ) const [inline], [virtual], [inherited]
```

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

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

7.13.2.5 void optix::APIObj::checkError(

```
    RTResult code,  
    Context context ) const [inline], [virtual], [inherited]
```

7.13.2.6 void optix::APIObj::checkErrorNoGetContext(

```
    RTResult code ) const [inline], [inherited]
```

7.13.2.7 void optix::CommandListObj::destroy() [inline], [virtual]

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

Implements [optix::DestroyableObj](#).

7.13.2.8 void optix::CommandListObj::execute() [inline]

Finalize the command list so that it can be called, later. See [rtCommandListFinalize](#).

7.13.2.9 void optix::CommandListObj::finalize() [inline]

Finalize the command list so that it can be called, later. See [rtCommandListFinalize](#).

7.13.2.10 RTcommandlist optix::CommandListObj::get() [inline]

Get the underlying OptiX C API RTcommandlist opaque pointer.

7.13.2.11 Context optix::CommandListObj::getContext() const [inline], [virtual]

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

Implements optix::APIObj.

7.13.2.12 Exception optix::APIObj::makeException(

RTresult code,

RTcontext context) [inline], [static], [inherited]

For backwards compatibility. Use `Exception::makeException` instead.

7.13.2.13 int optix::APIObj::removeReference() [inline], [inherited]

Decrement the reference count for this object.

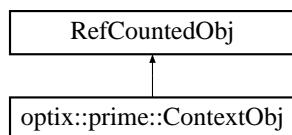
7.13.2.14 void optix::CommandListObj::validate() [inline], [virtual]

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

Implements optix::DestroyableObj.

7.14 optix::prime::ContextObj Class Reference

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 getLastErrorMessage ()`
- `RTPcontext getRTPcontext ()`

Static Public Member Functions

- static Context `create (RTPcontexttype type)`

7.14.1 Detailed Description

Wraps the OptiX Prime C API `RTPcontext` opaque type and its associated function set representing an OptiX Prime context.

7.14.2 Member Function Documentation

7.14.2.1 Context `optix::prime::ContextObj::create (RTPcontexttype type) [inline], [static]`

Creates a Context object. See [rtpContextCreate](#).

7.14.2.2 BufferDesc `optix::prime::ContextObj::createBufferDesc (RTPbufferformat format, RTPbuffertype type, void * buffer) [inline]`

Creates a BufferDesc object. See [rtpBufferDescCreate](#).

7.14.2.3 Model `optix::prime::ContextObj::createModel() [inline]`

Creates a Model object. See [rtpModelCreate](#).

7.14.2.4 std::string `optix::prime::ContextObj::getLastErrorMessage() [inline]`

Returns a string describing last error encountered. See [rtpContextGetLastErrorString](#).

7.14.2.5 RTPcontext `optix::prime::ContextObj::getRTPcontext() [inline]`

Returns the `RTPcontext` context stored within this object.

7.14.2.6 void `optix::prime::ContextObj::setCpuThreads (unsigned numThreads) [inline]`

Sets the number of CPU threads used by a CPU context. See [rtpContextSetCpuThreads](#).

7.14.2.7 void `optix::prime::ContextObj::setCudaDeviceNumbers (const std::vector< unsigned > & deviceNumbers) [inline]`

Sets the CUDA devices used by a context.

See [rtpContextSetCudaDeviceNumbers](#) Note that this distribution can be rather costly if the rays are stored in device memory though. For maximum efficiency it is recommended to only ever select one device per context.

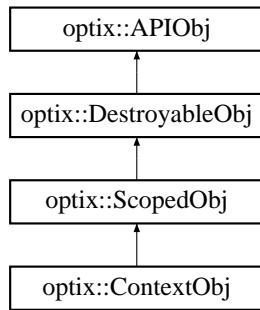
7.14.2.8 void `optix::prime::ContextObj::setCudaDeviceNumbers (unsigned deviceCount, const unsigned * deviceNumbers) [inline]`

Sets the CUDA devices used by a context.

See `rtpContextSetCudaDeviceNumbers`. Note that this distribution can be rather costly if the rays are stored in device memory though. For maximum efficiency it is recommended to only ever select one device per context.

7.15 optix::ContextObj Class Reference

Inheritance diagram for optix::ContextObj:



Public Member Functions

- `void destroy ()`
- `void validate ()`
- `Context getContext () const`
- `void compile ()`
- `int getRunningState () const`
- `RTcontext get ()`
- `void addReference ()`
- `int removeReference ()`
- `virtual void checkError (RTresult code, Context context) const`
- `void checkErrorNoGetContext (RTresult code) const`
- `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, RTgttarget target)
- Buffer getBufferFromId (int buffer_id)
- Program getProgramFromId (int program_id)
- TextureSampler getTextureSamplerFromId (int sampler_id)
- Geometry createGeometry ()
- GeometryTriangles createGeometryTriangles ()
- GeometryInstance createGeometryInstance ()
- template<class Iterator >
GeometryInstance createGeometryInstance (Geometry geometry, Iterator matlbegin, Iterator matlend)
- template<class Iterator >
GeometryInstance createGeometryInstance (GeometryTriangles geometrytriangles, Iterator matlbegin, Iterator matlend)
- GeometryInstance createGeometryInstance (GeometryTriangles geometry, Material mat)
- 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 &filename, const std::string &program_name)
- Program createProgramFromPTXFiles (const std::vector< std::string > &filenames, const std::string &program_name)
- Program createProgramFromPTXFiles (const std::vector< const char * > &filenames, const std::string &program_name)
- Program createProgramFromPTXString (const std::string &ptx, const std::string &program_name)
- Program createProgramFromPTXStrings (const std::vector< std::string > &ptxStrings, const std::string &program_name)

- `Program createProgramFromPTXStrings (const std::vector< const char * > &ptxStrings, const std::string &program_name)`
- `Selector createSelector ()`
- `TextureSampler createTextureSampler ()`

- `PostprocessingStage createBuiltinPostProcessingStage (const std::string &builtin_name)`
- `CommandList createCommandList ()`

- 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`
- `bool getPreferFastRecompiles () const`
- `bool getForceInlineUserFunctions () const`
- `int getGPUPagingActive () const`
- `int getGPUPagingForcedOff () const`
- `RTsize getAvailableDeviceMemory (int ordinal) const`

- `void setCPUNumThreads (int cpu_num_threads)`
- `void setPreferFastRecompiles (bool enabled)`
- `void setForceInlineUserFunctions (bool enabled)`
- `void setDiskCacheLocation (const std::string &path)`
- `std::string getDiskCacheLocation ()`
- `void setDiskCacheMemoryLimits (RTsize lowWaterMark, RTsize highWaterMark)`
- `void getDiskCacheMemoryLimits (RTsize &lowWaterMark, RTsize &highWaterMark)`
- `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 setMaxCallableProgramDepth (unsigned int max_depth)`
- `unsigned int getMaxCallableProgramDepth () const`
- `void setMaxTraceDepth (unsigned int max_depth)`
- `unsigned int getMaxTraceDepth () const`
- `void setTimeoutCallback (RTtimeoutcallback callback, double min_polling_seconds)`
- `void setUsageReportCallback (RTusagereportcallback callback, int verbosity, void *cbdata)`
- `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 std::string getDevicePCIBusId (int ordinal)`
- `static void getDeviceAttribute (int ordinal, RTdeviceattribute attrib, RTsize size, void *p)`
- `static std::vector< int > getCompatibleDevices (int ordinal)`
- `static Context create ()`
- `static Exception makeException (RTresult code, RTcontext context)`

7.15.1 Detailed Description

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

7.15.2 Member Function Documentation

7.15.2.1 void optix::APIObj::addReference() [inline], [inherited]

Increment the reference count for this object.

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

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

See [APIObj::checkError](#)

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

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

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

Deprecated in OptiX 4.0 See [rtContextCompile](#)

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

Creates a Context object. See [rtContextCreate](#).

7.15.2.7 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](#).

7.15.2.8 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](#).

7.15.2.9 Acceleration optix::ContextObj::createAcceleration (

```
    const std::string & builder,  
    const std::string & ignored = "" ) [inline]
```

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

7.15.2.10 Buffer optix::ContextObj::createBuffer (

```
    unsigned int type ) [inline]
```

Create a buffer with given RTbuffertype. See [rtBufferCreate](#).

7.15.2.11 Buffer optix::ContextObj::createBuffer (

```
    unsigned int type,  
    RTformat format ) [inline]
```

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

7.15.2.12 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](#).

7.15.2.13 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](#).

7.15.2.14 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](#).

7.15.2.15 Buffer optix::ContextObj::createBufferForCUDA (

unsigned int *type*) [inline]

Create a buffer for CUDA with given RTbuffertype. See [rtBufferCreate](#).

7.15.2.16 Buffer optix::ContextObj::createBufferForCUDA (

unsigned int *type*,
RTformat *format*) [inline]

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

7.15.2.17 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](#).

7.15.2.18 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](#).

7.15.2.19 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](#).

7.15.2.20 Buffer optix::ContextObj::createBufferFromGLBO (

unsigned int *type*,
unsigned int *vbo*) [inline]

Create buffer from GL buffer object. See [rtBufferCreateFromGLBO](#).

7.15.2.21 PostprocessingStage optix::ContextObj::createBuiltinPostProcessingStage (

```
const std::string & builtin_name ) [inline]
```

Create a builtin postprocessing stage. See [rtPostProcessingStageCreateBuiltin](#).

7.15.2.22 CommandList optix::ContextObj::createCommandList() [inline]

Create a new command list. See [rtCommandListCreate](#).

```
7.15.2.23 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](#).

```
7.15.2.24 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](#).

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

See [rtGeometryCreate](#).

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

See [rtGeometryGroupCreate](#).

```
7.15.2.27 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](#)

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

See [rtGeometryInstanceCreate](#).

7.15.2.29 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](#)

7.15.2.30 template<class Iterator > GeometryInstance optix::ContextObj::createGeometryInstance (

GeometryTriangles *geometrytriangles*,

Iterator *matlbegin*,

Iterator *matlend*)

Create a geometry instance with a GeometryTriangles object and a set of associated materials.

See [rtGeometryInstanceCreate](#), [rtGeometryInstanceSetMaterialCount](#), and [rtGeometryInstanceSetMaterial](#)

7.15.2.31 GeometryInstance optix::ContextObj::createGeometryInstance (

GeometryTriangles *geometry*,

Material *mat*) [inline]

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

7.15.2.32 GeometryTriangles optix::ContextObj::createGeometryTriangles() [inline]

See [rtGeometryTrianglesCreate](#).

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

See [rtGroupCreate](#).

7.15.2.34 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](#)

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

See [rtMaterialCreate](#).

7.15.2.36 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](#), [rtBufferSetMipLevelCount](#), and [rtBufferSetSize1D](#).

7.15.2.37 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](#), [rtBufferSetMipLevelCount](#), and [rtBufferSetSize2D](#).

7.15.2.38 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](#), [rtBufferSetMipLevelCount](#), and [rtBufferSetSize3D](#).

7.15.2.39 Program optix::ContextObj::createProgramFromPTXFile (

```
const std::string & filename,  

const std::string & program_name ) [inline]
```

See [rtProgramCreateFromPTXFile](#).

7.15.2.40 Program optix::ContextObj::createProgramFromPTXFiles (

```
const std::vector< std::string > & filenames,  

const std::string & program_name ) [inline]
```

See [rtProgramCreateFromPTXFiles](#).

7.15.2.41 Program optix::ContextObj::createProgramFromPTXFiles (

```
const std::vector< const char * > & filenames,  

const std::string & program_name ) [inline]
```

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

7.15.2.42 Program optix::ContextObj::createProgramFromPTXString (

```
    const std::string & ptx,  
    const std::string & program_name ) [inline]
```

See [rtProgramCreateFromPTXString](#).

7.15.2.43 Program optix::ContextObj::createProgramFromPTXStrings (

```
    const std::vector< std::string > & ptxStrings,  
    const std::string & program_name ) [inline]
```

See [rtProgramCreateFromPTXStrings](#).

7.15.2.44 Program optix::ContextObj::createProgramFromPTXStrings (

```
    const std::vector< const char * > & ptxStrings,  
    const std::string & program_name ) [inline]
```

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

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

See [rtSelectorCreate](#).

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

See [rtTextureSamplerCreate](#).

7.15.2.47 TextureSampler optix::ContextObj::createTextureSamplerFromGLImage (

```
    unsigned int id,  
    RTgltarget target ) [inline]
```

Create TextureSampler from GL image. See [rtTextureSamplerCreateFromGLImage](#).

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

See [rtTransformCreate](#).

7.15.2.49 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](#).

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

Destroy Context and all of its associated objects. See [rtContextDestroy](#).

Implements [optix::DestroyableObj](#).

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

Return the OptiX C API RTcontext object.

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

See [rtContextGetAttribute](#).

7.15.2.53 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](#).

7.15.2.54 std::vector< int > optix::ContextObj::getCompatibleDevices(int *ordinal*) [inline], [static]

Call [rtDeviceGetAttribute](#) and return the list of ordinals compatible with the device; a device is always compatible with itself.

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

Retrieve the Context object associated with this APIObject.

In this case, simply returns itself.

Implements [optix::APIObj](#).

7.15.2.56 int optix::ContextObj::getCPUNumThreads() const [inline]

See [rtContextGetAttribute](#).

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

Call [rtDeviceGetAttribute](#) and return the desired attribute value.

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

Call [rtDeviceGetDeviceCount](#) and returns number of valid devices.

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

Call [rtDeviceGetAttribute](#) and return the name of the device.

7.15.2.60 std::string optix::ContextObj::getDevicePCIBusId(

int *ordinal*) [inline], [static]

Call `rtDeviceGetAttribute` and return the PCI bus id of the device.

7.15.2.61 std::string optix::ContextObj::getDiskCacheLocation() const [inline]

See `rtContextGetAttribute`.

7.15.2.62 void optix::ContextObj::getDiskCacheMemoryLimits(RTsize & *lowWaterMark*, RTsize & *highWaterMark*) [inline]

See `rtContextGetAttribute`.

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

See `rtContextGetDeviceCount`.

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

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

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

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

See `rtContextGetEntryPointCount`.

7.15.2.66 std::string optix::ContextObj::getErrorString(RTResult *code*) const [inline]

See `rtContextGetErrorString`.

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

See `rtContextGetExceptionEnabled`.

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

See `rtContextGetExceptionProgram`.

7.15.2.69 bool optix::ContextObj::getForceInlineUserFunctions() const [inline]

See `rtContextGetAttribute`.

7.15.2.70 int optix::ContextObj::getGPU Paging Active() const [inline]

Deprecated in OptiX 4.0 See `rtContextGetAttribute`

7.15.2.71 int optix::ContextObj::getGPU Paging Forced Off() const [inline]

Deprecated in OptiX 4.0 See [rtContextGetAttribute](#)

7.15.2.72 unsigned int optix::ContextObj::getMaxCallableProgramDepth() const [inline]

See [rtContextGetMaxCallableProgramDepth](#).

7.15.2.73 int optix::ContextObj::getMaxTextureCount() const [inline]

See [rtContextGetAttribute](#)

7.15.2.74 unsigned int optix::ContextObj::getMaxTraceDepth() const [inline]

See [rtContextGetMaxTraceDepth](#).

7.15.2.75 Program optix::ContextObj::getMissProgram(unsigned int ray_type_index) const [inline]

See [rtContextGetMissProgram](#).

7.15.2.76 bool optix::ContextObj::getPreferFastRecompiles() const [inline]

See [rtContextGetAttribute](#).

7.15.2.77 RTsize optix::ContextObj::getPrintBufferSize() const [inline]

See [rtContextGetPrintBufferSize](#).

7.15.2.78 bool optix::ContextObj::getPrintEnabled() const [inline]

See [rtContextGetPrintEnabled](#).

7.15.2.79 optix::int3 optix::ContextObj::getPrintLaunchIndex() const [inline]

See [rtContextGetPrintLaunchIndex](#).

7.15.2.80 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](#).

7.15.2.81 Program optix::ContextObj::getRayGenerationProgram(unsigned int entry_point_index) const [inline]

See [rtContextGetRayGenerationProgram](#).

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

See [rtContextGetRayTypeCount](#).

7.15.2.83 int optix::ContextObj::getRunningState() const [inline]

See [rtContextGetRunningState](#).

7.15.2.84 RTsize optix::ContextObj::getStackSize() const [inline]

See [rtContextGetStackSize](#).

7.15.2.85 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](#).

7.15.2.86 RTsize optix::ContextObj::getUsedHostMemory() const [inline]

See [rtContextGetAttribute](#).

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

Query variable by index. See [rt\[ObjectType\]GetVariable](#).

Implements [optix::ScopedObj](#).

7.15.2.88 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](#).

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

See [rtContextLaunch](#)

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

See [rtContextLaunch](#).

7.15.2.91 void optix::ContextObj::launch (unsigned int *entry_point_index*, RTsize *image_width*,

```
RTsize image_height,  

RTsize image_depth ) [inline]
```

See [rtContextLaunch](#).

7.15.2.92 void optix::ContextObj::launchProgressive (

```
unsigned int entry_point_index,  

RTsize image_width,  

RTsize image_height,  

unsigned int max_subframes ) [inline]
```

See [rtContextLaunchProgressive](#)

7.15.2.93 Exception optix::APIObj::makeException (

```
RTresult code,  

RTcontext context ) [inline], [static], [inherited]
```

For backwards compatibility. Use [Exception::makeException](#) instead.

7.15.2.94 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](#).

7.15.2.95 int optix::APIObj::removeReference () [inline], [inherited]

Decrement the reference count for this object.

7.15.2.96 void optix::ContextObj::removeVariable (

```
Variable v ) [inline], [virtual]
```

Remove a variable associated with this object.

Implements [optix::ScopedObj](#).

7.15.2.97 template<class T > void optix::ContextObj::setAttribute (

```
RTcontextattribute attribute,  

const T & val ) [inline]
```

See [rtContextSetAttribute](#).

7.15.2.98 void optix::ContextObj::setCPUNumThreads (

```
int cpu_num_threads ) [inline]
```

See [rtContextSetAttribute](#)

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

See [rtContextSetDevices](#)

```
7.15.2.100 void optix::ContextObj::setDiskCacheLocation (
    const std::string & path ) [inline]
```

See [rtContextSetAttribute](#).

```
7.15.2.101 void optix::ContextObj::setDiskCacheMemoryLimits (
    RTsize lowWaterMark,
    RTsize highWaterMark ) [inline]
```

See [rtContextSetAttribute](#).

```
7.15.2.102 void optix::ContextObj::setEntryPointCount (
    unsigned int num_entry_points ) [inline]
```

See [rtContextSetEntryPointCount](#).

```
7.15.2.103 void optix::ContextObj::setExceptionEnabled (
    RTexception exception,
    bool enabled ) [inline]
```

See [rtContextSetExceptionEnabled](#).

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

See [rtContextSetExceptionProgram](#).

```
7.15.2.105 void optix::ContextObj::setForceInlineUserFunctions (
    bool enabled ) [inline]
```

See [rtContextSetAttribute](#).

```
7.15.2.106 void optix::ContextObj::setGPU Paging Forced Off (
    int gpu_paging_forced_off ) [inline]
```

Deprecated in OptiX 4.0 See [rtContextSetAttribute](#)

```
7.15.2.107 void optix::ContextObj::setMaxCallableProgramDepth (
    unsigned int max_depth ) [inline]
```

See [rtContextSetMaxCallableProgramDepth](#)

7.15.2.108 void optix::ContextObj::setMaxTraceDepth (
 unsigned int *max_depth*) [inline]

See [rtContextSetMaxTraceDepth](#)

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

See [rtContextSetMissProgram](#).

7.15.2.110 void optix::ContextObj::setPreferFastRecompiles (
 bool *enabled*) [inline]

See [rtContextSetAttribute](#).

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

See [rtContextSetPrintBufferSize](#).

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

See [rtContextSetPrintEnabled](#)

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

See [rtContextSetPrintLaunchIndex](#).

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

See [rtContextSetRayGenerationProgram](#)

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

See [rtContextSetRayTypeCount](#).

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

See [rtContextSetStackSize](#)

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

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

7.15.2.118 void optix::ContextObj::setUsageReportCallback (
 RTusagereportcallback *callback*,
 int *verbosity*,
 void * *cbdata*) [inline]

See [rtContextSetUsageReportCallback](#) RTusagereportcallback is defined as typedef void (RTusagereportcallback)(int, const char, const char*, void*).

7.15.2.119 void optix::ContextObj::stopProgressive () [inline]

See [rtContextStopProgressive](#).

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

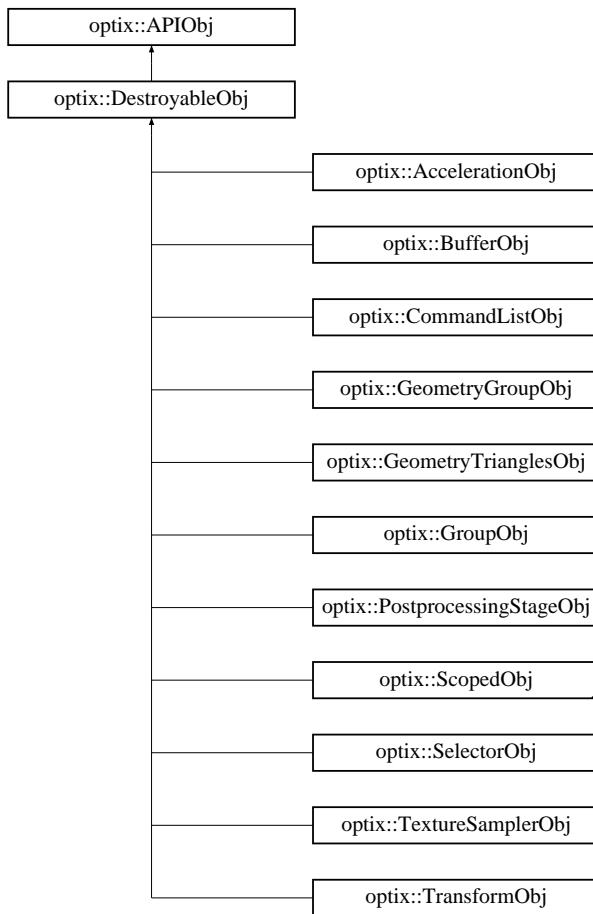
See [rtContextValidate](#).

Implements optix::DestroyableObj.

7.16 rti_internal_callableprogram::CPArgVoid Class Reference

7.17 optix::DestroyableObj Class Reference

Inheritance diagram for optix::DestroyableObj:



Public Member Functions

- `virtual ~DestroyableObj ()`
- `virtual void destroy ()=0`
- `virtual void validate ()=0`
- `void addReference ()`
- `int removeReference ()`
- `virtual Context getContext () const =0`
- `virtual void checkError (RTresult code) const`
- `virtual void checkError (RTresult code, Context context) const`
- `void checkErrorNoGetContext (RTresult code) const`

Static Public Member Functions

- `static Exception makeException (RTresult code, RTcontext context)`

7.17.1 Detailed Description

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

Wraps:

- RTcontext
- RTgeometry
- RTgeometrytriangles
- RTgeometryinstance
- RTgeometrygroup
- RTgroup
- RTmaterial
- RTprogram
- RTselector
- RTtexturesampler
- RTtransform

7.17.2 Constructor & Destructor Documentation

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

7.17.3 Member Function Documentation

7.17.3.1 void optix::APIObj::addReference() [inline], [inherited]

Increment the reference count for this object.

7.17.3.2 void optix::APIObj::checkError(

RTResult code) const [inline], [virtual], [inherited]

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

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

7.17.3.3 void optix::APIObj::checkError(

RTResult code,

Context context) const [inline], [virtual], [inherited]

7.17.3.4 void optix::APIObj::checkErrorNoGetContext(

RTResult code) const [inline], [inherited]

7.17.3.5 virtual void optix::DestroyableObj::destroy() [pure virtual]

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

Implemented in [optix::CommandListObj](#), [optix::PostprocessingStageObj](#), [optix::BufferObj](#), [optix::TextureSamplerObj](#), [optix::MaterialObj](#), [optix::GeometryTrianglesObj](#), [optix::GeometryObj](#), [optix::GeometryInstanceObj](#), [optix::AccelerationObj](#), [optix::SelectorObj](#), [optix::TransformObj](#), [optix::GeometryGroupObj](#), [optix::GroupObj](#), [optix::ProgramObj](#), and [optix::ContextObj](#).

7.17.3.6 virtual Context optix::APIObj::getContext() const [pure virtual], [inherited]

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

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

7.17.3.7 Exception `optix::APIObj::makeException` (

`RTResult code,`
`RTcontext context) [inline], [static], [inherited]`

For backwards compatibility. Use `Exception::makeException` instead.

7.17.3.8 int `optix::APIObj::removeReference` () [inline], [inherited]

Decrement the reference count for this object.

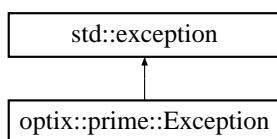
7.17.3.9 virtual void `optix::DestroyableObj::validate` () [pure virtual]

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

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

7.18 optix::prime::Exception Class Reference

Inheritance diagram for `optix::prime::Exception`:



Public Member Functions

- `virtual ~Exception () throw ()`
- `RTPresResult getErrorCode () const`
- `const std::string & getErrorString () const`
- `virtual const char * what () const throw ()`
- `T what (T...args)`

Static Public Member Functions

- `static Exception makeException (RTPresResult code)`
- `static Exception makeException (RTPresResult code, RTPcontext context)`

7.18.1 Detailed Description

Encapsulates an OptiX Prime exception.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 `virtual optix::prime::Exception::~Exception() throw() [inline], [virtual]`

7.18.3 Member Function Documentation

7.18.3.1 `RTPresult optix::prime::Exception::getErrorCode() const [inline]`

Stores the RTPresult error code for this exception.

7.18.3.2 `const std::string & optix::prime::Exception::getErrorString() const [inline]`

Stores the human-readable error string associated with this exception.

7.18.3.3 `Exception optix::prime::Exception::makeException(RTPresult code) [inline], [static]`

Returns a string describing last error encountered. See [rtpGetErrorString](#).

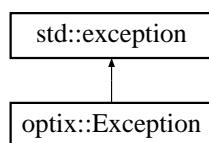
7.18.3.4 `Exception optix::prime::Exception::makeException(RTPresult code, RTPcontext context) [inline], [static]`

Returns a string describing last error encountered. See [rtpContextGetLastErrorString](#).

7.18.3.5 `const char * optix::prime::Exception::what() const throw() [inline], [virtual]`

7.19 optix::Exception Class Reference

Inheritance diagram for optix::Exception:



Public Member Functions

- `Exception (const std::string &message, RTPresult error_code=RT_ERROR_UNKNOWN)`
- `virtual ~Exception () throw()`
- `const std::string & getErrorString () const`
- `RTPresult getErrorCode () const`
- `virtual const char * what () const throw()`
- `T what (T...args)`

Static Public Member Functions

- static `Exception makeException (RTresult code, RTcontext context)`

7.19.1 Detailed Description

`Exception` class for error reporting from the OptiXpp API.

Encapsulates an error message, often the direct result of a failed OptiX C API function call and subsequent `rtContextGetErrorString` call.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 `optix::Exception::Exception (`
 `const std::string & message,`
 `RTresult error_code = RT_ERROR_UNKNOWN) [inline]`

Create exception.

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

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

7.19.3 Member Function Documentation

7.19.3.1 `RTResult optix::Exception::getErrorCode () const [inline]`

Retrieve the error code.

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

Retrieve the error message.

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

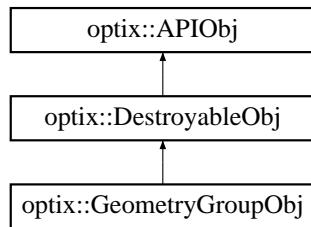
Helper for creating exceptions from an `RTResult` code origination from an OptiX C API function call.

7.19.3.4 `virtual const char* optix::Exception::what () const throw () [inline], [virtual]`

From `std::exception`.

7.20 optix::GeometryGroupObj Class Reference

Inheritance diagram for `optix::GeometryGroupObj`:



Public Member Functions

- void `destroy ()`
- void `validate ()`
- `Context getContext () const`
- `RTgeometrygroup get ()`
- void `addReference ()`
- int `removeReference ()`
- virtual void `checkError (RTresult code) const`
- virtual void `checkError (RTresult code, Context context) const`
- void `checkErrorNoGetContext (RTresult code) const`

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

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

- void `setFlags (RTinstanceflags flags)`
- `RTinstanceflags getFlags () const`
- void `setVisibilityMask (RTvisibilitymask mask)`
- `RTvisibilitymask getVisibilityMask () const`

Static Public Member Functions

- static `Exception makeException (RTresult code, RTcontext context)`

7.20.1 Detailed Description

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

7.20.2 Member Function Documentation

7.20.2.1 `unsigned int optix::GeometryGroupObj::addChild (GeometryInstance child) [inline]`

Set a new child in this group and return its new index. See [rtGeometryGroupSetChild](#).

7.20.2.2 `void optix::APIObj::addReference () [inline], [inherited]`

Increment the reference count for this object.

7.20.2.3 `void optix::APIObj::checkError (RTResult code) const [inline], [virtual], [inherited]`

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

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

7.20.2.4 `void optix::APIObj::checkError (RTResult code, Context context) const [inline], [virtual], [inherited]`

7.20.2.5 `void optix::APIObj::checkErrorNoGetContext (RTResult code) const [inline], [inherited]`

7.20.2.6 `void optix::GeometryGroupObj::destroy () [inline], [virtual]`

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

Implements [optix::DestroyableObj](#).

7.20.2.7 `RTgeometrygroup optix::GeometryGroupObj::get () [inline]`

Get the underlying OptiX C API RTgeometrygroup opaque pointer.

7.20.2.8 `Acceleration optix::GeometryGroupObj::getAcceleration () const [inline]`

Query the Acceleration structure for this group. See [rtGeometryGroupGetAcceleration](#).

7.20.2.9 `GeometryInstance optix::GeometryGroupObj::getChild (unsigned int index) const [inline]`

Query an indexed GeometryInstance within this group. See [rtGeometryGroupGetChild](#).

7.20.2.10 `unsigned int optix::GeometryGroupObj::getChildCount () const [inline]`

Query the number of children for this group. See [rtGeometryGroupGetChildCount](#).

7.20.2.11 `unsigned int optix::GeometryGroupObj::getChildIndex () const [inline]`

GeometryInstance *child*) const [inline]

Query a child in this group for its index. See [rtGeometryGroupGetChild](#).

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

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).

Implements [optix::APIObj](#).

7.20.2.13 RTinstanceflags optix::GeometryGroupObj::getFlags() const [inline]

See [rtGeometryGroupSetFlags](#)

7.20.2.14 RTvisibilitymask optix::GeometryGroupObj::getVisibilityMask() const [inline]

See [rtGeometryGroupSetFlags](#)

7.20.2.15 Exception optix::APIObj::makeException(

RTresult *code*,
RTcontext *context*) [inline], [static], [inherited]

For backwards compatibility. Use [Exception::makeException](#) instead.

7.20.2.16 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.

7.20.2.17 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.

7.20.2.18 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.

7.20.2.19 int optix::APIObj::removeReference() [inline], [inherited]

Decrement the reference count for this object.

7.20.2.20 void optix::GeometryGroupObj::setAcceleration(

Acceleration *acceleration*) [inline]

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

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

Set an indexed GeometryInstance child of this group. See [rtGeometryGroupSetChild](#).

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

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

7.20.2.23 void optix::GeometryGroupObj::setFlags (
RTinstanceflags *flags*) [inline]

See [rtGeometryGroupSetFlags](#)

7.20.2.24 void optix::GeometryGroupObj::setVisibilityMask (
RTvisibilitymask *mask*) [inline]

See [rtGeometryGroupSetVisibilityMask](#).

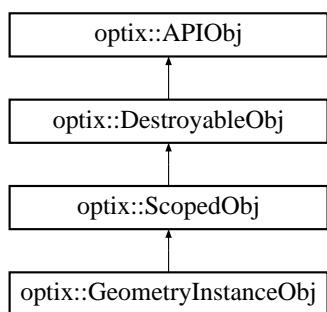
7.20.2.25 void optix::GeometryGroupObj::validate() [inline], [virtual]

call [rt\[ObjectType\]Validate](#) on the underlying OptiX C object

Implements [optix::DestroyableObj](#).

7.21 optix::GeometryInstanceObj Class Reference

Inheritance diagram for [optix::GeometryInstanceObj](#):



Public Member Functions

- void [destroy\(\)](#)
- void [validate\(\)](#)

- `Context getContext () const`
- `RTgeometryinstance get ()`
- `void addReference ()`
- `int removeReference ()`
- `virtual void checkError (RTresult code) const`
- `virtual void checkError (RTresult code, Context context) const`
- `void checkErrorNoGetContext (RTresult code) const`

- `void setGeometry (Geometry geometry)`
- `Geometry getGeometry () const`
- `void setGeometryTriangles (GeometryTriangles geometry)`
- `GeometryTriangles getGeometryTriangles () 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`

Static Public Member Functions

- static `Exception makeException (RTresult code, RTcontext context)`

7.21.1 Detailed Description

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

7.21.2 Member Function Documentation

7.21.2.1 `unsigned int optix::GeometryInstanceObj::addMaterial (` `Material material) [inline]`

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

7.21.2.2 `void optix::APIObj::addReference () [inline], [inherited]`

Increment the reference count for this object.

7.21.2.3 void optix::APIObj::checkError (
 RTresult code) const [inline], [virtual], [inherited]

Check the given result code and throw an error with appropriate message if the code is not RTsuccess.
Reimplemented in [optix::ContextObj](#).

7.21.2.4 void optix::APIObj::checkError (
 RTresult code,
 Context context) const [inline], [virtual], [inherited]

7.21.2.5 void optix::APIObj::checkErrorNoGetContext (
 RTresult code) const [inline], [inherited]

7.21.2.6 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](#).

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

call [rt\[ObjectType\]Destroy](#) on the underlying OptiX C object

Implements [optix::DestroyableObj](#).

7.21.2.8 RTgeometryinstance optix::GeometryInstanceObj::get () [inline]

Get the underlying OptiX C API RTgeometryinstance opaque pointer.

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

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).

Implements [optix::APIObj](#).

7.21.2.10 Geometry optix::GeometryInstanceObj::getGeometry () const [inline]

Get the geometry object associated with this instance. See [rtGeometryInstanceGetGeometry](#).

7.21.2.11 GeometryTriangles optix::GeometryInstanceObj::getGeometryTriangles () const [inline]

Get the GeometryTriangles object associated with this instance. See [rtGeometryInstanceGetGeometryTriangles](#).

7.21.2.12 Material optix::GeometryInstanceObj::getMaterial (

unsigned int *idx*) const [inline]

Get the material at given index. See [rtGeometryInstanceGetMaterial](#).

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

Query the number of materials associated with this instance. See [rtGeometryInstanceGetMaterialCount](#).

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

Query variable by index. See [rt\[ObjectType\]GetVariable](#).

Implements [optix::ScopedObj](#).

7.21.2.15 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](#).

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

For backwards compatibility. Use [Exception::makeException](#) instead.

7.21.2.17 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](#).

7.21.2.18 int optix::APIObj::removeReference() [inline], [inherited]

Decrement the reference count for this object.

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

Remove a variable associated with this object.

Implements [optix::ScopedObj](#).

7.21.2.20 void optix::GeometryInstanceObj::setGeometry(

Geometry *geometry*) [inline]

Set the geometry object associated with this instance. See [rtGeometryInstanceSetGeometry](#).

7.21.2.21 void optix::GeometryInstanceObj::setGeometryTriangles (
GeometryTriangles *geometry*) [inline]

Set the GeometryTriangles object associated with this instance. See [rtGeometryInstanceSetGeometryTriangles](#).

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

Set the material at given index. See [rtGeometryInstanceSetMaterial](#).

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

Set the number of materials associated with this instance. See [rtGeometryInstanceSetMaterialCount](#).

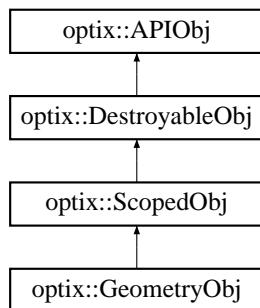
7.21.2.24 void optix::GeometryInstanceObj::validate () [inline], [virtual]

call [rt\[ObjectType\]Validate](#) on the underlying OptiX C object

Implements [optix::DestroyableObj](#).

7.22 optix::GeometryObj Class Reference

Inheritance diagram for [optix::GeometryObj](#):



Public Member Functions

- [void destroy \(\)](#)
- [void validate \(\)](#)
- [Context getContext \(\) const](#)
- [RTgeometry get \(\)](#)
- [void addReference \(\)](#)
- [int removeReference \(\)](#)

- virtual void `checkError (RTresult code) const`
- virtual void `checkError (RTresult code, Context context) const`
- void `checkErrorNoGetContext (RTresult code) const`

- 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 `setMotionRange (float timeBegin, float timeEnd)`
- void `getMotionRange (float &timeBegin, float &timeEnd)`
- void `setMotionBorderMode (RTmotionbordermode beginMode, RTmotionbordermode endMode)`
- void `getMotionBorderMode (RTmotionbordermode &beginMode, RTmotionbordermode &endMode)`
- void `setMotionSteps (unsigned int n)`
- unsigned int `getMotionSteps ()`

- 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`

- void `setFlags (RTgeometryflags flags)`
- RTgeometryflags `setFlags () const`

Static Public Member Functions

- static Exception `makeException (RTresult code, RTcontext context)`

7.22.1 Detailed Description

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

7.22.2 Member Function Documentation

7.22.2.1 void optix::APIObj::addReference () [inline], [inherited]

Increment the reference count for this object.

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

Check the given result code and throw an error with appropriate message if the code is not RTsuccess.
Reimplemented in [optix::ContextObj](#).

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

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

7.22.2.5 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](#).

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

call [rt\[ObjectType\]Destroy](#) on the underlying OptiX C object

Implements [optix::DestroyableObj](#).

7.22.2.7 RTgeometry optix::GeometryObj::get () [inline]

Get the underlying OptiX C API RTgeometry opaque pointer.

7.22.2.8 Program optix::GeometryObj::getBoundingBoxProgram () const [inline]

Get the bounding box program for this geometry. See [rtGeometryGetBoundingBoxProgram](#).

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

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).

Implements [optix::APIObj](#).

7.22.2.10 Program optix::GeometryObj::getIntersectionProgram () const [inline]

Get the intersection program for this geometry. See [rtGeometryGetIntersectionProgram](#).

7.22.2.11 void optix::GeometryObj::getMotionBorderMode (
 RTmotionbordermode & *beginMode*,
 RTmotionbordermode & *endMode*) [inline]

Query the motion border mode for this geometry object.

See [rtGeometryGetMotionBorderMode](#)

7.22.2.12 void optix::GeometryObj::getMotionRange (
 float & *timeBegin*,
 float & *timeEnd*) [inline]

Query the motion time range for this geometry object.

See [rtGeometryGetMotionRange](#)

7.22.2.13 unsigned int optix::GeometryObj::getMotionSteps () const [inline]

Query the number of motion steps for this geometry object.

See [rtGeometryGetMotionSteps](#)

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

Query the number of primitives in this geometry object (eg, number of triangles in mesh).

See [rtGeometryGetPrimitiveCount](#)

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

Query the primitive index offset for this geometry object.

See [rtGeometryGetPrimitiveIndexOffset](#)

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

Query variable by index. See [rt\[ObjectType\]GetVariable](#).

Implements [optix::ScopedObj](#).

7.22.2.17 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](#).

7.22.2.18 bool optix::GeometryObj::isDirty () const [inline]

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

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

For backwards compatibility. Use [Exception::makeException](#) instead.

7.22.2.20 void optix::GeometryObj::markDirty() [inline]

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

7.22.2.21 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](#).

7.22.2.22 int optix::APIObj::removeReference() [inline], [inherited]

Decrement the reference count for this object.

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

Remove a variable associated with this object.

Implements [optix::ScopedObj](#).

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

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

7.22.2.25 void optix::GeometryObj::setFlags (
RTgeometryflags *flags*) [inline]

See [rtGeometrySetFlags](#)

7.22.2.26 RTgeometryflags optix::GeometryObj::setFlags() const [inline]

See [rtGeometrySetFlags](#)

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

Set the intersection program for this geometry. See [rtGeometrySetIntersectionProgram](#).

7.22.2.28 void optix::GeometryObj::setMotionBorderMode (
RTmotionbordermode *beginMode*,
RTmotionbordermode *endMode*) [inline]

Set motion border mode for this geometry object.

See [rtGeometrySetMotionBorderMode](#)

7.22.2.29 void optix::GeometryObj::setMotionRange (

```
float timeBegin,
float timeEnd ) [inline]
```

Set motion time range for this geometry object. See [rtGeometrySetMotionRange](#)

7.22.2.30 void optix::GeometryObj::setMotionSteps (
unsigned int n) [inline]

Set the number of motion steps for this geometry object.

See [rtGeometrySetMotionSteps](#)

7.22.2.31 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](#)

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

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

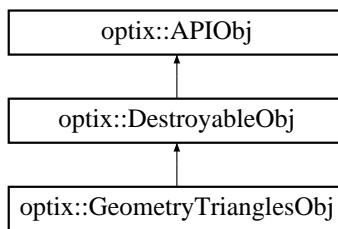
7.22.2.33 void optix::GeometryObj::validate() [inline], [virtual]

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

Implements [optix::DestroyableObj](#).

7.23 optix::GeometryTrianglesObj Class Reference

Inheritance diagram for `optix::GeometryTrianglesObj`:



Public Member Functions

- `void destroy()`
- `void validate()`
- `Context getContext() const`
- `RTgeometrytriangles get()`
- `void addReference()`
- `int removeReference()`

- virtual void `checkError (RTresult code)` const
- virtual void `checkError (RTresult code, Context context)` const
- void `checkErrorNoGetContext (RTresult code)` const
- void `setPrimitiveIndexOffset (unsigned int index_offset)`
- unsigned int `getPrimitiveIndexOffset ()` const
- void `setPrimitiveCount (unsigned int num_triangles)`
- unsigned int `getPrimitiveCount ()` const
- void `setPreTransformMatrix (bool transpose, const float *matrix)`
- void `getPreTransformMatrix (bool transpose, float *matrix)`
- void `setTriangleIndices (Buffer index_buffer, RTformat tri_indices_format)`
- void `setTriangleIndices (Buffer index_buffer, RTsize index_buffer_byte_offset, RTformat tri_indices_format)`
- void `setTriangleIndices (Buffer index_buffer, RTsize index_buffer_byte_offset, RTsize tri_indices_byte_stride, RTformat tri_indices_format)`
- void `setVertices (unsigned int num_vertices, Buffer vertex_buffer, RTformat position_format)`
- void `setVertices (unsigned int num_vertices, Buffer vertex_buffer, RTsize vertex_buffer_byte_offset, RTformat position_format)`
- void `setVertices (unsigned int num_vertices, Buffer vertex_buffer, RTsize vertex_buffer_byte_offset, RTsize vertex_byte_stride, RTformat position_format)`
- void `setAttributeProgram (Program program)`
- Program `getAttributeProgram ()` 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
- void `setMotionRange (float timeBegin, float timeEnd)`
- void `getMotionRange (float &timeBegin, float &timeEnd)` const
- void `setMotionBorderMode (RTmotionbordermode beginMode, RTmotionbordermode endMode)`
- void `getMotionBorderMode (RTmotionbordermode &beginMode, RTmotionbordermode &endMode)` const
- void `setMotionSteps (unsigned int n)`
- unsigned int `getMotionSteps ()` const
- void `setMotionVertices (unsigned int num_vertices, Buffer vertex_buffer, RTsize vertex_buffer_byte_offset, RTsize vertex_byte_stride, RTsize vertex_motion_step_byte_stride, RTformat position_format)`
- template<class BufferIterator >
void `setMotionVerticesMultiBuffer (unsigned int num_vertices, BufferIterator vertex_buffers_begin, BufferIterator vertex_buffers_end, RTsize vertex_buffer_byte_offset, RTsize vertex_byte_stride, RTformat position_format)`
- void `setBuildFlags (RTgeometrybuildflags build_flags)`
- void `setMaterialCount (unsigned int num_materials)`

- `unsigned int getMaterialCount () const`
- `void setMaterialIndices (Buffer material_index_buffer, RTsize material_index_buffer_byte_offset, RTsize material_index_byte_stride, RTformat material_index_format)`
- `void setFlagsPerMaterial (unsigned int material_index, RTgeometryflags flags)`
- `RTgeometryflags getFlagsPerMaterial (unsigned int material_index) const`

Static Public Member Functions

- `static Exception makeException (RTresult code, RTcontext context)`

7.23.1 Detailed Description

GeometryTriangles wraps the OptiX C API RTgeometrytriangles opaque type and its associated function set.

7.23.2 Member Function Documentation

7.23.2.1 void optix::APIObj::addReference () [inline], [inherited]

Increment the reference count for this object.

7.23.2.2 void optix::APIObj::checkError (

`RTresult code) const [inline], [virtual], [inherited]`

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

Reimplemented in `optix::ContextObj`.

7.23.2.3 void optix::APIObj::checkError (

`RTresult code,`

`Context context) const [inline], [virtual], [inherited]`

7.23.2.4 void optix::APIObj::checkErrorNoGetContext (

`RTresult code) const [inline], [inherited]`

7.23.2.5 Variable optix::GeometryTrianglesObj::declareVariable (

`const std::string & name) [inline]`

See `rtGeometryTrianglesSetTriangleIndices`

7.23.2.6 void optix::GeometryTrianglesObj::destroy () [inline], [virtual]

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

Implements `optix::DestroyableObj`.

7.23.2.7 RTgeometrytriangles optix::GeometryTrianglesObj::get () [inline]

Get the underlying OptiX C API RTgeometrytriangles opaque pointer.

7.23.2.8 Program optix::GeometryTrianglesObj::getAttributeProgram() const [inline]

Get the attribute program for this GeometryTriangles object. See [rtGeometryTrianglesGetAttributeProgram](#).

7.23.2.9 Context optix::GeometryTrianglesObj::getContext() const [inline], [virtual]

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).

Implements [optix::APIObj](#).

7.23.2.10 RTgeometryflags optix::GeometryTrianglesObj::getFlagsPerMaterial(unsigned int material_index) const [inline]

See [rtGeometryTrianglesSetBuildFlags](#)

7.23.2.11 unsigned int optix::GeometryTrianglesObj::getMaterialCount() const [inline]

See [rtGeometryTrianglesGetMaterialCount](#).

7.23.2.12 void optix::GeometryTrianglesObj::getMotionBorderMode(RTmotionbordermode & beginMode, RTmotionbordermode & endMode) const [inline]

Query the motion border mode for this geometry triangles object.

See [rtGeometryTrianglesGetMotionBorderMode](#)

7.23.2.13 void optix::GeometryTrianglesObj::getMotionRange(float & timeBegin, float & timeEnd) const [inline]

Query the motion time range for this geometry triangles object.

See [rtGeometryTrianglesGetMotionRange](#)

7.23.2.14 unsigned int optix::GeometryTrianglesObj::getMotionSteps() const [inline]

Query the number of motion steps for this geometry triangles object.

See [rtGeometryTrianglesGetMotionSteps](#)

7.23.2.15 void optix::GeometryTrianglesObj::getPreTransformMatrix(bool transpose, float * matrix) [inline]

7.23.2.16 unsigned int optix::GeometryTrianglesObj::getPrimitiveCount() const [inline]

Query the number of triangles in this geometry triangles object.

See [rtGeometryTrianglesGetPrimitiveCount](#)

7.23.2.17 `unsigned int optix::GeometryTrianglesObj::getPrimitiveIndexOffset() const [inline]`

Query the primitive index offset for this GeometryTriangles object.

See [rtGeometryTrianglesGetPrimitiveIndexOffset](#)

7.23.2.18 Variable `optix::GeometryTrianglesObj::getVariable(unsigned int index) const [inline]`

See [rtGeometryTrianglesSetTriangleIndices](#)

7.23.2.19 `unsigned int optix::GeometryTrianglesObj::getVariableCount() const [inline]`

See [rtGeometryTrianglesSetTriangleIndices](#)

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

For backwards compatibility. Use [Exception::makeException](#) instead.

7.23.2.21 Variable `optix::GeometryTrianglesObj::queryVariable(const std::string & name) const [inline]`

See [rtGeometryTrianglesSetTriangleIndices](#)

7.23.2.22 `int optix::APIObj::removeReference() [inline], [inherited]`

Decrement the reference count for this object.

7.23.2.23 `void optix::GeometryTrianglesObj::removeVariable(Variable v) [inline]`

See [rtGeometryTrianglesSetTriangleIndices](#)

7.23.2.24 `void optix::GeometryTrianglesObj::setAttributeProgram(Program program) [inline]`

Set the attribute program for this GeometryTriangles object. See [rtGeometryTrianglesSetAttributeProgram](#).

7.23.2.25 `void optix::GeometryTrianglesObj::setBuildFlags(RTgeometrybuildflags build_flags) [inline]`

See [rtGeometryTrianglesSetBuildFlags](#)

7.23.2.26 `void optix::GeometryTrianglesObj::setFlagsPerMaterial(unsigned int material_index,`

RTgeometryflags *flags*) [inline]

See [rtGeometryTrianglesSetFlagsPerMaterial](#).

7.23.2.27 void optix::GeometryTrianglesObj::setMaterialCount (
 unsigned int *num_materials*) [inline]

See [rtGeometryTrianglesSetMaterialCount](#).

7.23.2.28 void optix::GeometryTrianglesObj::setMaterialIndices (
 Buffer *material_index_buffer*,
 RTsize *material_index_buffer_byte_offset*,
 RTsize *material_index_byte_stride*,
 RTformat *material_index_format*) [inline]

See [rtGeometryTrianglesSetMaterialIndices](#).

7.23.2.29 void optix::GeometryTrianglesObj::setMotionBorderMode (
 RTmotionbordermode *beginMode*,
 RTmotionbordermode *endMode*) [inline]

Set motion border mode for this geometry triangles object.

See [rtGeometryTrianglesSetMotionBorderMode](#)

7.23.2.30 void optix::GeometryTrianglesObj::setMotionRange (
 float *timeBegin*,
 float *timeEnd*) [inline]

Set motion time range for this geometry triangles object. See [rtGeometryTrianglesSetMotionRange](#)

7.23.2.31 void optix::GeometryTrianglesObj::setMotionSteps (
 unsigned int *n*) [inline]

Set the number of motion steps for this geometry triangles object.

See [rtGeometryTrianglesSetMotionSteps](#)

7.23.2.32 void optix::GeometryTrianglesObj::setMotionVertices (
 unsigned int *num_vertices*,
 Buffer *vertex_buffer*,
 RTsize *vertex_buffer_byte_offset*,
 RTsize *vertex_byte_stride*,
 RTsize *vertex_motion_step_byte_stride*,
 RTformat *position_format*) [inline]

See [rtGeometryTrianglesSetMotionVertices](#).

```
7.23.2.33 template<class BufferIterator > void op-
tix::GeometryTrianglesObj::setMotionVerticesMultiBuffer
(
    unsigned int num_vertices,
    BufferIterator vertex_buffers_begin,
    BufferIterator vertex_buffers_end,
    RTsize vertex_buffer_byte_offset,
    RTsize vertex_byte_stride,
    RTformat position_format ) [inline]
```

See [rtGeometryTrianglesSetMotionVerticesMultiBuffer](#).

```
7.23.2.34 void optix::GeometryTrianglesObj::setPreTransformMatrix (
    bool transpose,
    const float * matrix ) [inline]
```

```
7.23.2.35 void optix::GeometryTrianglesObj::setPrimitiveCount (
    unsigned int num_triangles ) [inline]
```

Set the number of triangles in this geometry triangles object.

See [rtGeometryTrianglesSetPrimitiveCount](#)

```
7.23.2.36 void optix::GeometryTrianglesObj::setPrimitiveIndexOffset (
    unsigned int index_offset ) [inline]
```

Set the primitive index offset for this GeometryTriangles object. See [rtGeometryTrianglesSetPrimitiveIndexOffset](#)

```
7.23.2.37 void optix::GeometryTrianglesObj::setTriangleIndices (
    Buffer index_buffer,
    RTformat tri_indices_format ) [inline]
```

See [rtGeometryTrianglesSetTriangleIndices](#)

```
7.23.2.38 void optix::GeometryTrianglesObj::setTriangleIndices (
    Buffer index_buffer,
    RTsize index_buffer_byte_offset,
    RTformat tri_indices_format ) [inline]
```

See [rtGeometryTrianglesSetTriangleIndices](#)

```
7.23.2.39 void optix::GeometryTrianglesObj::setTriangleIndices (
    Buffer index_buffer,
    RTsize index_buffer_byte_offset,
    RTsize tri_indices_byte_stride,
```

RTformat *tri_indices_format*) [inline]

See [rtGeometryTrianglesSetTriangleIndices](#)

7.23.2.40 void optix::GeometryTrianglesObj::setVertices (
unsigned int *num_vertices*,
Buffer *vertex_buffer*,
RTformat *position_format*) [inline]

See [rtGeometryTrianglesSetVertices](#).

7.23.2.41 void optix::GeometryTrianglesObj::setVertices (
unsigned int *num_vertices*,
Buffer *vertex_buffer*,
RTsize *vertex_buffer_byte_offset*,
RTformat *position_format*) [inline]

See [rtGeometryTrianglesSetTriangleIndices](#)

7.23.2.42 void optix::GeometryTrianglesObj::setVertices (
unsigned int *num_vertices*,
Buffer *vertex_buffer*,
RTsize *vertex_buffer_byte_offset*,
RTsize *vertex_byte_stride*,
RTformat *position_format*) [inline]

See [rtGeometryTrianglesSetTriangleIndices](#)

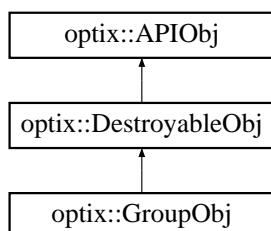
7.23.2.43 void optix::GeometryTrianglesObj::validate() [inline], [virtual]

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

Implements `optix::DestroyableObj`.

7.24 optix::GroupObj Class Reference

Inheritance diagram for `optix::GroupObj`:



Public Member Functions

- void `destroy ()`
- void `validate ()`
- `Context getContext () const`
- `RTgroup get ()`
- void `addReference ()`
- int `removeReference ()`
- virtual void `checkError (RTresult code) const`
- virtual void `checkError (RTresult code, Context context) const`
- void `checkErrorNoGetContext (RTresult code) const`

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

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

- void `setVisibilityMask (RTvisibilitymask)`
- `RTvisibilitymask getVisibilityMask () const`

Static Public Member Functions

- static `Exception makeException (RTresult code, RTcontext context)`

7.24.1 Detailed Description

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

7.24.2 Member Function Documentation

7.24.2.1 template<typename T > unsigned int optix::GroupObj::addChild (*T child*) [inline]

Set a new child in this group and returns its new index. See `rtGroupSetChild`.

7.24.2.2 void optix::APIObj::addReference() [inline], [inherited]

Increment the reference count for this object.

7.24.2.3 void optix::APIObj::checkError (RTResult code) const [inline], [virtual], [inherited]

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

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

7.24.2.4 void optix::APIObj::checkError (RTResult code, Context context) const [inline], [virtual], [inherited]**7.24.2.5 void optix::APIObj::checkErrorNoGetContext (RTResult code) const [inline], [inherited]****7.24.2.6 void optix::GroupObj::destroy() [inline], [virtual]**

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

Implements [optix::DestroyableObj](#).

7.24.2.7 RTgroup optix::GroupObj::get() [inline]

Get the underlying OptiX C API RTgroup opaque pointer.

7.24.2.8 Acceleration optix::GroupObj::getAcceleration() const [inline]

Query the Acceleration structure for this group. See [rtGroupGetAcceleration](#).

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

Query an indexed child within this group. See [rtGroupGetChild](#).

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

Query the number of children for this group. See [rtGroupGetChildCount](#).

7.24.2.11 template<typename T > unsigned int optix::GroupObj::getChildIndex (T child) const [inline]

Query a child in this group for its index. See [rtGroupGetChild](#).

7.24.2.12 RTobjecttype optix::GroupObj::getChildType (unsigned int index) const [inline]

Query indexed child's type. See [rtGroupGetChildType](#).

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

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).

Implements [optix::APIObj](#).

7.24.2.14 RTvisibilitymask optix::GroupObj::getVisibilityMask() const [inline]

See [rtGroupSetVisibilityMask](#)

7.24.2.15 Exception optix::APIObj::makeException(

RTresult *code*,

RTcontext *context*) [inline], [static], [inherited]

For backwards compatibility. Use [Exception::makeException](#) instead.

7.24.2.16 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.

7.24.2.17 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.

7.24.2.18 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.

7.24.2.19 int optix::APIObj::removeReference() [inline], [inherited]

Decrement the reference count for this object.

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

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

7.24.2.21 template<typename T > void optix::GroupObj::setChild(unsigned int *index*,

T child) [inline]

Set an indexed child within this group. See [rtGroupSetChild](#).

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

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

7.24.2.23 void optix::GroupObj::setVisibilityMask (
RTvisibilitymask mask) [inline]

See [rtGroupSetVisibilityMask](#)

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

call [rt\[ObjectType\]Validate](#) on the underlying OptiX C object

Implements [optix::DestroyableObj](#).

7.25 optix::Handle< T > Class Template Reference

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 \(RTobject p\)](#)
- [static Handle< T > create \(\)](#)
- [static unsigned int getDeviceCount \(\)](#)

7.25.1 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.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 template<class T> optix::Handle< T >::Handle() [inline]

Default constructor initializes handle to null pointer.

7.25.2.2 template<class T> optix::Handle< T >::Handle(T * ptr) [inline]

Takes a raw pointer to an API object and creates a handle.

7.25.2.3 template<class T> template<class U > optix::Handle< T >::Handle(U * ptr) [inline]

Takes a raw pointer of arbitrary type and creates a handle.

7.25.2.4 template<class T> optix::Handle< T >::Handle(const Handle< T > & copy) [inline]

Takes a handle of the same type and creates a handle.

7.25.2.5 template<class T> template<class U > optix::Handle< T >::Handle(const Handle< U > & copy) [inline]

Takes a handle of some other type and creates a handle.

7.25.2.6 template<class T> optix::Handle< T >::~Handle() [inline]

Decrements reference count on the handled object.

7.25.3 Member Function Documentation

7.25.3.1 template<class T> static Handle<T> optix::Handle< T >::create() [inline], [static]

Static object creation. Only valid for contexts.

7.25.3.2 template<class T> T* optix::Handle< T >::get() [inline]

Retrieve the handled object.

7.25.3.3 template<class T> const T* optix::Handle< T >::get() const [inline]

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

Query the machine device count. Only valid for contexts.

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

implicit bool cast based on NULLness of wrapped pointer

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

Dereferences the handle.

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

7.25.3.8 template<class T> Handle<T>& optix::Handle< T >::operator=(const Handle< T > & copy) [inline]

Assignment of handle with same underlying object type.

7.25.3.9 template<class T> template<class U> Handle<T>& optix::Handle< T >::operator=(const Handle< U > & copy) [inline]

Assignment of handle with different underlying object type.

7.25.3.10]

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

Variable access operator.

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

7.25.3.11]

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

Variable access operator.

Identical to [operator\[\]\(const std::string& varname\)](#)

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

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

can be interpreted as either

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

or

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

7.25.3.12 template<class T> static Handle<T> optix::Handle< T >::take (typename T::api_t p) [inline], [static]

Takes a base optix api opaque type and creates a handle to optixpp wrapper type.

7.25.3.13 template<class T> static Handle<T> optix::Handle< T >::take (RTObject p) [inline], [static]

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

7.26 rti_internal_callableprogram::is_CPAVoid< T1 > Struct Template Reference

Static Public Attributes

- static const bool result = false

7.26.1 Member Data Documentation

7.26.1.1 template<typename T1> const bool rti_internal_callableprogram::is_CPAVoid< T1 >::result = false [static]

7.27 rti_internal_callableprogram::is_CPAVoid< CPAVoid > Struct Template Reference

Static Public Attributes

- static const bool result = true

7.27.1 Member Data Documentation

7.27.1.1 const bool rti_internal_callableprogram::is_CPAVoid< CPAVoid >::result = true [static]

7.28 optix::markedCallableProgramId< T > Class Template Reference

Public Member Functions

- markedCallableProgramId ()
- markedCallableProgramId (int)

7.28.1 Constructor & Destructor Documentation

7.28.1.1 `template<typename T> optix::markedCallableProgramId< T >::markedCallableProgramId() [inline]`

7.28.1.2 `template<typename T> optix::markedCallableProgramId< T >::markedCallableProgramId(int) [inline]`

7.29 **rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T > Class Template Reference**

Public Member Functions

- `__device__ __forceinline__ markedCallableProgramIdBase(int id, const char *callSiteName)`
- `__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`
- `const char * m_callSiteName`

7.29.1 Constructor & Destructor Documentation

7.29.1.1 `template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT,
Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T
>::markedCallableProgramIdBase (`

`int id,`

const char * *callSiteName*) [inline], [explicit]

7.29.2 Member Function Documentation

**7.29.2.1 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T,
Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() () [inline]**

**7.29.2.2 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T,
Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (**

Arg0T arg0) [inline]

7.29.2.3 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T,
Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator()
Arg0T arg0,
Arg1T arg1) [inline]

7.29.2.4 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T,
Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator()
Arg0T arg0,
Arg1T arg1,

Arg2T *arg2*) [inline]

7.29.2.5 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T,
Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (
Arg0T *arg0*,
Arg1T *arg1*,
Arg2T *arg2*,
Arg3T *arg3*) [inline]

7.29.2.6 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T,
Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (
Arg0T *arg0*,
Arg1T *arg1*,
Arg2T *arg2*,
Arg3T *arg3*,

Arg4T arg4) [inline]

7.29.2.7 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T,
Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (

Arg0T arg0,
Arg1T arg1,
Arg2T arg2,
Arg3T arg3,
Arg4T arg4,
Arg5T arg5) [inline]

7.29.2.8 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T,
Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (

Arg0T arg0,
Arg1T arg1,
Arg2T arg2,
Arg3T arg3,
Arg4T arg4,
Arg5T arg5,

Arg6T *arg6*) [inline]

7.29.2.9 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T,
Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (
Arg0T *arg0*,
Arg1T *arg1*,
Arg2T *arg2*,
Arg3T *arg3*,
Arg4T *arg4*,
Arg5T *arg5*,
Arg6T *arg6*,
Arg7T *arg7*) [inline]

7.29.2.10 template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T =
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T,
Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (
Arg0T *arg0*,
Arg1T *arg1*,
Arg2T *arg2*,
Arg3T *arg3*,
Arg4T *arg4*,

```
Arg5T arg5,  
Arg6T arg6,  
Arg7T arg7,  
Arg8T arg8 ) [inline]
```

```
7.29.2.11 template<typename ReturnT , typename Arg0T =  
rti_internal_callableprogram::CPArgVoid, typename Arg1T =  
rti_internal_callableprogram::CPArgVoid, typename Arg2T =  
rti_internal_callableprogram::CPArgVoid, typename Arg3T =  
rti_internal_callableprogram::CPArgVoid, typename Arg4T =  
rti_internal_callableprogram::CPArgVoid, typename Arg5T =  
rti_internal_callableprogram::CPArgVoid, typename Arg6T =  
rti_internal_callableprogram::CPArgVoid, typename Arg7T =  
rti_internal_callableprogram::CPArgVoid, typename Arg8T =  
rti_internal_callableprogram::CPArgVoid, typename Arg9T =  
rti_internal_callableprogram::CPArgVoid> __device__ __forceinline__ ReturnT  
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T,  
Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::operator() (  
Arg0T arg0,  
Arg1T arg1,  
Arg2T arg2,  
Arg3T arg3,  
Arg4T arg4,  
Arg5T arg5,  
Arg6T arg6,  
Arg7T arg7,  
Arg8T arg8,
```

Arg9T *arg9*) [inline]

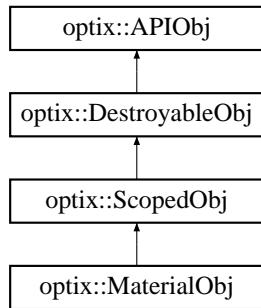
7.29.3 Member Data Documentation

7.29.3.1 **template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T =
rti_internal_callableprogram::CPArgVoid, typename Arg2T =
rti_internal_callableprogram::CPArgVoid, typename Arg3T =
rti_internal_callableprogram::CPArgVoid, typename Arg4T =
rti_internal_callableprogram::CPArgVoid, typename Arg5T =
rti_internal_callableprogram::CPArgVoid, typename Arg6T =
rti_internal_callableprogram::CPArgVoid, typename Arg7T =
rti_internal_callableprogram::CPArgVoid, typename Arg8T =
rti_internal_callableprogram::CPArgVoid, typename Arg9T
= rti_internal_callableprogram::CPArgVoid> const char*
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T,
Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::m_callSiteName
[protected]**

7.29.3.2 **template<typename ReturnT , typename Arg0T =
rti_internal_callableprogram::CPArgVoid, typename Arg1T
= rti_internal_callableprogram::CPArgVoid, typename
Arg2T = rti_internal_callableprogram::CPArgVoid, typename
Arg3T = rti_internal_callableprogram::CPArgVoid, typename
Arg4T = rti_internal_callableprogram::CPArgVoid, typename
Arg5T = rti_internal_callableprogram::CPArgVoid, typename
Arg6T = rti_internal_callableprogram::CPArgVoid, typename
Arg7T = rti_internal_callableprogram::CPArgVoid, typename
Arg8T = rti_internal_callableprogram::CPArgVoid, type-
name Arg9T = rti_internal_callableprogram::CPArgVoid> int
rti_internal_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T,
Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T >::m_id [protected]**

7.30 optix::MaterialObj Class Reference

Inheritance diagram for optix::MaterialObj:



Public Member Functions

- void `destroy ()`
- void `validate ()`
- `Context getContext () const`
- `RTmaterial get ()`
- void `addReference ()`
- int `removeReference ()`
- virtual void `checkError (RTresult code) const`
- virtual void `checkError (RTresult code, Context context) const`
- void `checkErrorNoGetContext (RTresult code) const`

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

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

Static Public Member Functions

- static `Exception makeException (RTresult code, RTcontext context)`

7.30.1 Detailed Description

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

7.30.2 Member Function Documentation

7.30.2.1 void optix::APIObj::addReference () [inline], [inherited]

Increment the reference count for this object.

7.30.2.2 void optix::APIObj::checkError (
 RTresult code) const [inline], [virtual], [inherited]

Check the given result code and throw an error with appropriate message if the code is not RTsuccess.
Reimplemented in [optix::ContextObj](#).

7.30.2.3 void optix::APIObj::checkError (
 RTresult code,
 Context context) const [inline], [virtual], [inherited]

7.30.2.4 void optix::APIObj::checkErrorNoGetContext (
 RTresult code) const [inline], [inherited]

7.30.2.5 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](#).

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

call [rt\[ObjectType\]Destroy](#) on the underlying OptiX C object

Implements [optix::DestroyableObj](#).

7.30.2.7 RTmaterial optix::MaterialObj::get () [inline]

Get the underlying OptiX C API RTmaterial opaque pointer.

7.30.2.8 Program optix::MaterialObj::getAnyHitProgram (
 unsigned int ray_type_index) const [inline]

Get any hit program for this material at the given *ray_type* index. See [rtMaterialGetAnyHitProgram](#).

7.30.2.9 Program optix::MaterialObj::getClosestHitProgram (
 unsigned int ray_type_index) const [inline]

Get closest hit program for this material at the given *ray_type* index. See [rtMaterialGetClosestHitProgram](#).

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

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).

Implements [optix::APIObj](#).

7.30.2.11 Variable optix::MaterialObj::getVariable (

unsigned int *index*) const [inline], [virtual]

Query variable by index. See [rt\[ObjectType\]GetVariable](#).

Implements [optix::ScopedObj](#).

7.30.2.12 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](#).

7.30.2.13 Exception optix::APIObj::makeException(

RTresult *code*,

RTcontext *context*) [inline], [static], [inherited]

For backwards compatibility. Use [Exception::makeException](#) instead.

7.30.2.14 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](#).

7.30.2.15 int optix::APIObj::removeReference() [inline], [inherited]

Decrement the reference count for this object.

7.30.2.16 void optix::MaterialObj::removeVariable(

Variable *v*) [inline], [virtual]

Remove a variable associated with this object.

Implements [optix::ScopedObj](#).

7.30.2.17 void optix::MaterialObj::setAnyHitProgram(

unsigned int *ray_type_index*,

Program *program*) [inline]

Set any hit program for this material at the given *ray_type* index. See [rtMaterialSetAnyHitProgram](#).

7.30.2.18 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](#).

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

call [rt\[ObjectType\]Validate](#) on the underlying OptiX C object

Implements [optix::DestroyableObj](#).

7.31 optix::Matrix< M, N > Class Template Reference

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 ()

7.31.1 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**

7.31.2 Member Typedef Documentation

7.31.2.1 template<unsigned int M, unsigned int N> typedef VectorDim<M>::VectorType optix::Matrix< M, N >::floatM

A row of the matrix.

**7.31.2.2 template<unsigned int M, unsigned int N> typedef VectorDim<N>::VectorType
optix::Matrix< M, N >::floatN**

7.31.3 Constructor & Destructor Documentation

**7.31.3.1 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

**7.31.3.2 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.

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

Copy the matrix.

7.31.4 Member Function Documentation

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

Returns the determinant of the matrix.

**7.31.4.2 template<> OPTIXU_INLINE RT_HOSTDEVICE float optix::Matrix< 3, 3 >::det()
const**

**7.31.4.3 template<> OPTIXU_INLINE RT_HOSTDEVICE float optix::Matrix< 4, 4 >::det()
const**

**7.31.4.4 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.

**7.31.4.5 template<> OPTIXU_INLINE RT_HOSTDEVICE Matrix< 4, 4 > optix::Matrix< 4, 4
>::fromBasis(**

```
const float3 & u,
const float3 & v,
const float3 & w,
const float3 & c )
```

7.31.4.6 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

7.31.4.7 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.

7.31.4.8 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.

7.31.4.9 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

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

Returns the identity matrix.

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

Returns the inverse of the matrix.

7.31.4.12 template<> OPTIXU_INLINE RT_HOSTDEVICE Matrix< 4, 4 > optix::Matrix< 4, 4 >::inverse () const

7.31.4.13 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.

7.31.4.14 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.

7.31.4.15]

```
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.

7.31.4.16]

```
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.

7.31.4.17 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.

7.31.4.18 template<> OPTIXU_INLINE RT_HOSTDEVICE Matrix< 4, 4 > optix::Matrix< 4, 4
>::rotate (
const float radians,
const float3 & axis)

7.31.4.19 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.

7.31.4.20 template<> OPTIXU_INLINE RT_HOSTDEVICE Matrix< 4, 4 > optix::Matrix< 4, 4
>::scale (
const float3 & vec)

7.31.4.21 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

```
7.31.4.22 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

```
7.31.4.23 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.

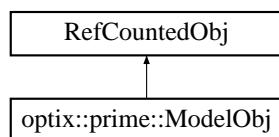
```
7.31.4.24 template<> OPTIXU_INLINE RT_HOSTDEVICE Matrix< 4, 4 > optix::Matrix< 4, 4
>::translate (
    const float3 & vec )
```

```
7.31.4.25 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.

7.32 optix::prime::ModelObj Class Reference

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, RTPbuffertype 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, RTPbufferformat 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 ()`

7.32.1 Detailed Description

Encapsulates an OptiX Prime model.

The purpose of a model is to represent a set of triangles and an acceleration structure.

7.32.2 Member Function Documentation

7.32.2.1 void optix::prime::ModelObj::copy (const Model & *srcModel*) [inline]

Copies one model to another. See [rtpModelCopy](#).

7.32.2.2 Query optix::prime::ModelObj::createQuery (RTPquerytype *queryType*) [inline]

Creates a Query object. See [rtpQueryCreate](#).

7.32.2.3 void optix::prime::ModelObj::finish () [inline]

Blocks current thread until model update is finished. See [rtpModelFinish](#).

7.32.2.4 Context optix::prime::ModelObj::getContext () [inline]

Returns the context associated within this object.

7.32.2.5 RTPmodel optix::prime::ModelObj::getRTPmodel () [inline]

Returns the RTPmodel model stored within this object.

7.32.2.6 int optix::prime::ModelObj::isFinished () [inline]

Polls the status of a model update. See [rtpModelGetFinished](#).

7.32.2.7 void optix::prime::ModelObj::setBuilderParameter (RTPbuilderparam *param*, RTPsize *size*,

```
const void * p ) [inline]
```

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

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

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

7.32.2.9 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

7.32.2.10 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

7.32.2.11 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

7.32.2.12 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

7.32.2.13 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

7.32.2.14 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

7.32.2.15 void optix::prime::ModelObj::update (
unsigned hints) [inline]

Creates the acceleration structure over the triangles. See [rtpModelUpdate](#).

7.33 optix::Onb Struct Reference

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](#)

7.33.1 Detailed Description

Orthonormal basis.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 OPTIXU_INLINE RT_HOSTDEVICE optix::Onb::Onb (
const float3 & *normal*) [inline]

7.33.3 Member Function Documentation

7.33.3.1 OPTIXU_INLINE RT_HOSTDEVICE void optix::Onb::inverse_transform (
float3 & *p*) const [inline]

7.33.4 Member Data Documentation

7.33.4.1 float3 optix::Onb::m_binormal

7.33.4.2 float3 optix::Onb::m_normal

7.33.4.3 float3 optix::Onb::m_tangent

7.34 OptiX API Struct Reference

Public Attributes

- unsigned int [version](#)
- [PRTACCELERATIONCREATE](#) [rtAccelerationCreate](#)
- [PRTACCELERATIONDESTROY](#) [rtAccelerationDestroy](#)
- [PRTACCELERATIONGETBUILDER](#) [rtAccelerationGetBuilder](#)
- [PRTACCELERATIONGETCONTEXT](#) [rtAccelerationGetContext](#)
- [PRTACCELERATIONGETDATA](#) [rtAccelerationGetData](#)
- [PRTACCELERATIONGETDATASIZE](#) [rtAccelerationGetDataSize](#)
- [PRTACCELERATIONGETPROPERTY](#) [rtAccelerationGetProperty](#)
- [PRTACCELERATIONGETTRaverser](#) [rtAccelerationGetTraverser](#)
- [PRTACCELERATIONISDIRTY](#) [rtAccelerationIsDirty](#)
- [PRTACCELERATIONMARKDIRTY](#) [rtAccelerationMarkDirty](#)
- [PRTACCELERATIONSETBUILDER](#) [rtAccelerationSetBuilder](#)
- [PRTACCELERATIONSETDATA](#) [rtAccelerationSetData](#)
- [PRTACCELERATIONsetProperty](#) [rtAccelerationSetProperty](#)
- [PRTACCELERATIONSETTRaverser](#) [rtAccelerationSetTraverser](#)
- [PRTACCELERATIONVALIDATE](#) [rtAccelerationValidate](#)
- [PRTBUFFERCREATE](#) [rtBufferCreate](#)
- [PRTBUFFERCREATEFORCUDA](#) [rtBufferCreateForCUDA](#)
- [PRTBUFFERCREATEFROMGLBO](#) [rtBufferCreateFromGLBO](#)
- [PRTBUFFERDESTROY](#) [rtBufferDestroy](#)
- [PRTBUFFERGLREGISTER](#) [rtBufferGLRegister](#)
- [PRTBUFFERGLUNREGISTER](#) [rtBufferGLUnregister](#)
- [PRTBUFFERGETCONTEXT](#) [rtBufferGetContext](#)
- [PRTBUFFERGETDEVICEPOINTER](#) [rtBufferGetDevicePointer](#)
- [PRTBUFFERGETDIMENSIONALITY](#) [rtBufferGetDimensionality](#)

- PRTBUFFERGETELEMENTSIZE rtBufferGetElementSize
- PRTBUFFERGETFORMAT rtBufferGetFormat
- PRTBUFFERGETGLBOID rtBufferGetGLBOId
- PRTBUFFERGETID rtBufferGetId
- PRTBUFFERGETMIPLEVELCOUNT rtBufferGetMipLevelCount
- PRTBUFFERGETMIPLEVELSIZE1D rtBufferGetMipLevelSize1D
- PRTBUFFERGETMIPLEVELSIZE2D rtBufferGetMipLevelSize2D
- PRTBUFFERGETMIPLEVELSIZE3D rtBufferGetMipLevelSize3D
- PRTBUFFERGETSIZE1D rtBufferGetSize1D
- PRTBUFFERGETSIZE2D rtBufferGetSize2D
- PRTBUFFERGETSIZE3D rtBufferGetSize3D
- PRTBUFFERGETSIZEV rtBufferGetSizev
- PRTBUFFERMAP rtBufferMap
- PRTBUFFERMAPEX rtBufferMapEx
- PRTBUFFERMARKDIRTY rtBufferMarkDirty
- PRTBUFFERSETDEVICEPOINTER rtBufferSetDevicePointer
- PRTBUFFERSETELEMENTSIZE rtBufferSetElementSize
- PRTBUFFERSETFORMAT rtBufferSetFormat
- PRTBUFFERSETMIPLEVELCOUNT rtBufferSetMipLevelCount
- PRTBUFFERSETSIZE1D rtBufferSetSize1D
- PRTBUFFERSETSIZE2D rtBufferSetSize2D
- PRTBUFFERSETSIZE3D rtBufferSetSize3D
- PRTBUFFERSETSIZEV rtBufferSetSizev
- PRTBUFFERUNMAP rtBufferUnmap
- PRTBUFFERUNMAPEX rtBufferUnmapEx
- PRTBUFFERVALIDATE rtBufferValidate
- PRTBUFFERGETPROGRESSIVEUPDATEREADY rtBufferGetProgressiveUpdateReady
- PRTBUFFERBINDPROGRESSIVESTREAM rtBufferBindProgressiveStream
- PRTBUFFERSETATTRIBUTE rtBufferSetAttribute
- PRTBUFFERGETATTRIBUTE rtBufferGetAttribute
- PRTCMDLISTCREATE rtCommandListCreate
- PRTCMDLISTDESTROY rtCommandListDestroy
- PRTCMDLISTAPPENDPOSTPROCESSINGSTAGE
 rtCommandListAppendPostprocessingStage
- PRTCMDLISTAPPENDLAUNCH2D rtCommandListAppendLaunch2D
- PRTCMDLISTFINALIZE rtCommandListFinalize
- PRTCMDLISTEXECUTE rtCommandListExecute
- PRTCMDLISTGETCONTEXT rtCommandListGetContext
- PRTEXECUTECOMPILER rtContextCompile
- PRTEXECUTECREATE rtContextCreate
- PRTEXECUTDECLAREVARIABLE rtContextDeclareVariable
- PRTEXECUTDESTROY rtContextDestroy
- PRTEXECUTSETATTRIBUTE rtContextSetAttribute
- PRTEXECUTGETATTRIBUTE rtContextGetAttribute
- PRTEXECUTGETBUFFERFROMID rtContextGetBufferFromId
- PRTEXECUTGETDEVICECOUNT rtContextGetDeviceCount

- PRTCONTEXTGETDEVICES rtContextGetDevices
- PRTCONTEXTGETENTRYPOINTCOUNT rtContextGetEntryPointCount
- PRTCONTEXTGETERRORSTRING rtContextGetErrorString
- PRTCONTEXTGETEXCEPTIONENABLED rtContextGetExceptionEnabled
- PRTCONTEXTGETEXCEPTIONPROGRAM rtContextGetExceptionProgram
- PRTCONTEXTGETMISSPROGRAM rtContextGetMissProgram
- PRTCONTEXTGETPRINTBUFFERSIZE rtContextGetPrintBufferSize
- PRTCONTEXTGETPRINTENABLED rtContextGetPrintEnabled
- PRTCONTEXTGETPRINTLAUNCHINDEX rtContextGetPrintLaunchIndex
- PRTCONTEXTGETPROGRAMFROMID rtContextGetProgramFromId
- PRTCONTEXTGETRAYGENERATIONPROGRAM rtContextGetRayGenerationProgram
- PRTCONTEXTGETRAYTYPECOUNT rtContextGetRayTypeCount
- PRTCONTEXTGETRUNNINGSTATE rtContextGetRunningState
- PRTCONTEXTGETSTACKSIZE rtContextGetStackSize
- PRTCONTEXTGETTEXTURESAMPLERFROMID rtContextGetTextureSamplerFromId
- PRTCONTEXTGETVARIABLE rtContextGetVariable
- PRTCONTEXTGETVARIABLECOUNT rtContextGetVariableCount
- PRTCONTEXTLAUNCH1D rtContextLaunch1D
- PRTCONTEXTLAUNCH2D rtContextLaunch2D
- PRTCONTEXTLAUNCH3D rtContextLaunch3D
- PRTCONTEXTLAUNCHPROGRESSIVE2D rtContextLaunchProgressive2D
- PRTCONTEXTSTOPPROGRESSIVE rtContextStopProgressive
- PRTCONTEXTQUERYVARIABLE rtContextQueryVariable
- PRTCONTEXTREMOVEVARIABLE rtContextRemoveVariable
- PRTCONTEXTSETDEVICES rtContextSetDevices
- PRTCONTEXTSETENTRYPOINTCOUNT rtContextSetEntryPointCount
- PRTCONTEXTSETEXCEPTIONENABLED rtContextSetExceptionEnabled
- PRTCONTEXTSETEXCEPTIONPROGRAM rtContextSetExceptionProgram
- PRTCONTEXTSETMISSPROGRAM rtContextSetMissProgram
- PRTCONTEXTSETPRINTBUFFERSIZE rtContextSetPrintBufferSize
- PRTCONTEXTSETPRINTENABLED rtContextSetPrintEnabled
- PRTCONTEXTSETPRINTLAUNCHINDEX rtContextSetPrintLaunchIndex
- PRTCONTEXTSETRAYGENERATIONPROGRAM rtContextSetRayGenerationProgram
- PRTCONTEXTSETRAYTYPECOUNT rtContextSetRayTypeCount
- PRTDEPRECATEDAPIFUNCTION rtContextSetRemoteDevice
- PRTCONTEXTSETSTACKSIZE rtContextSetStackSize
- PRTCONTEXTSETTIMEOUTCALLBACK rtContextSetTimeoutCallback
- PRTCONTEXTSETUSAGEREPORTCALLBACK rtContextSetUsageReportCallback
- PRTCONTEXTVALIDATE rtContextValidate
- PRTDEVICEGETATTRIBUTE rtDeviceGetAttribute
- PRTDEVICEGETDEVICECOUNT rtDeviceGetDeviceCount
- PRTGEOMETRYCREATE rtGeometryCreate
- PRTGEOMETRYDECLARE VARIABLE rtGeometryDeclareVariable
- PRTGEOMETRYDESTROY rtGeometryDestroy
- PRTGEOMETRYGETBOUNDINGBOXPROGRAM rtGeometryGetBoundingBoxProgram
- PRTGEOMETRYGETCONTEXT rtGeometryGetContext

- PRTGEOMETRYGETINTERSECTIONPROGRAM rtGeometryGetIntersectionProgram
- PRTGEOMETRYGETPRIMITIVECOUNT rtGeometryGetPrimitiveCount
- PRTGEOMETRYGETPRIMITIVEINDEXOFFSET rtGeometryGetPrimitiveIndexOffset
- PRTGEOMETRYGETVARIABLE rtGeometryGetVariable
- PRTGEOMETRYGETVARIABLECOUNT rtGeometryGetVariableCount
- PRTGEOMETRYGROUPCREATE rtGeometryGroupCreate
- PRTGEOMETRYGROUPDESTROY rtGeometryGroupDestroy
- PRTGEOMETRYGROUPGETACCELERATION rtGeometryGroupGetAcceleration
- PRTGEOMETRYGROUPGETCHILD rtGeometryGroupGetChild
- PRTGEOMETRYGROUPGETCHILDCOUNT rtGeometryGroupGetChildCount
- PRTGEOMETRYGROUPGETCONTEXT rtGeometryGroupGetContext
- PRTGEOMETRYGROUPSETACCELERATION rtGeometryGroupSetAcceleration
- PRTGEOMETRYGROUPSETCHILD rtGeometryGroupSetChild
- PRTGEOMETRYGROUPSETCHILDCOUNT rtGeometryGroupSetChildCount
- PRTGEOMETRYGROUPVALIDATE rtGeometryGroupValidate
- PRTGEOMETRYINSTANCECREATE rtGeometryInstanceCreate
- PRTGEOMETRYINSTANCEDECLAREVARIABLE rtGeometryInstanceDeclareVariable
- PRTGEOMETRYINSTANCEDESTROY rtGeometryInstanceDestroy
- PRTGEOMETRYINSTANCEGETCONTEXT rtGeometryInstanceGetContext
- PRTGEOMETRYINSTANCEGETGEOMETRY rtGeometryInstanceGetGeometry
- PRTGEOMETRYINSTANCEGETMATERIAL rtGeometryInstanceGetMaterial
- PRTGEOMETRYINSTANCEGETMATERIALCOUNT rtGeometryInstanceGetMaterialCount
- PRTGEOMETRYINSTANCEGETVARIABLE rtGeometryInstanceGetVariable
- PRTGEOMETRYINSTANCEGETVARIABLECOUNT rtGeometryInstanceGetVariableCount
- PRTGEOMETRYINSTANCEQUERYVARIABLE rtGeometryInstanceQueryVariable
- PRTGEOMETRYINSTANCEREMOVEVARIABLE rtGeometryInstanceRemoveVariable
- PRTGEOMETRYINSTANCESETGEOMETRY rtGeometryInstanceSetGeometry
- PRTGEOMETRYINSTANCESETMATERIAL rtGeometryInstanceSetMaterial
- PRTGEOMETRYINSTANCESETMATERIALCOUNT rtGeometryInstanceSetMaterialCount
- PRTGEOMETRYINSTANCEVALIDATE rtGeometryInstanceValidate
- PRTGEOMETRYISDIRTY rtGeometryIsDirty
- PRTGEOMETRYMARKDIRTY rtGeometryMarkDirty
- PRTGEOMETRYQUERYVARIABLE rtGeometryQueryVariable
- PRTGEOMETRYREMOVEVARIABLE rtGeometryRemoveVariable
- PRTGEOMETRYSETBOUNDINGBOXPROGRAM rtGeometrySetBoundingBoxProgram
- PRTGEOMETRYSETINTERSECTIONPROGRAM rtGeometrySetIntersectionProgram
- PRTGEOMETRYSETPRIMITIVECOUNT rtGeometrySetPrimitiveCount
- PRTGEOMETRYSETPRIMITIVEINDEXOFFSET rtGeometrySetPrimitiveIndexOffset
- PRTGEOMETRYSETMOTIONRANGE rtGeometrySetMotionRange
- PRTGEOMETRYGETMOTIONRANGE rtGeometryGetMotionRange
- PRTGEOMETRYSETMOTIONBORDERMODE rtGeometrySetMotionBorderMode
- PRTGEOMETRYGETMOTIONBORDERMODE rtGeometryGetMotionBorderMode
- PRTGEOMETRYSETMOTIONSTEPS rtGeometrySetMotionSteps
- PRTGEOMETRYGETMOTIONSTEPS rtGeometryGetMotionSteps
- PRTGEOMETRYVALIDATE rtGeometryValidate
- PRTGETVERSION rtGetVersion

- PRTGLOBALSETATTRIBUTE rtGlobalSetAttribute
- PRTGLOBALGETATTRIBUTE rtGlobalGetAttribute
- PRTGROUPCREATE rtGroupCreate
- PRTGROUPDESTROY rtGroupDestroy
- PRTGROUPGETACCELERATION rtGroupGetAcceleration
- PRTGROUPGETCHILD rtGroupGetChild
- PRTGROUPGETCHILDCOUNT rtGroupGetChildCount
- PRTGROUPGETCHILDTYPE rtGroupGetChildType
- PRTGROUPGETCONTEXT rtGroupGetContext
- PRTGROUPSETACCELERATION rtGroupSetAcceleration
- PRTGROUPSETCHILD rtGroupSetChild
- PRTGROUPSETCHILDCOUNT rtGroupSetChildCount
- PRTGROUPVALIDATE rtGroupValidate
- PRTMATERIALCREATE rtMaterialCreate
- PRTMATERIALDECLAREVARIABLE rtMaterialDeclareVariable
- PRTMATERIALDESTROY rtMaterialDestroy
- PRTMATERIALGETANYHITPROGRAM rtMaterialGetAnyHitProgram
- PRTMATERIALGETCLOSESTHITPROGRAM rtMaterialGetClosestHitProgram
- PRTMATERIALGETCONTEXT rtMaterialGetContext
- PRTMATERIALGETVARIABLE rtMaterialGetVariable
- PRTMATERIALGETVARIABLECOUNT rtMaterialGetVariableCount
- PRTMATERIALQUERYVARIABLE rtMaterialQueryVariable
- PRTMATERIALREMOVEVARIABLE rtMaterialRemoveVariable
- PRTMATERIALSETANYHITPROGRAM rtMaterialSetAnyHitProgram
- PRTMATERIALSETCLOSESTHITPROGRAM rtMaterialSetClosestHitProgram
- PRTMATERIALVALIDATE rtMaterialValidate
- PRTPOSTPROCESSINGSTAGECREATEBUILTIN rtPostProcessingStageCreateBuiltin
- PRTPOSTPROCESSINGSTAGEDECLAREVARIABLE rtPostProcessingStageDeclareVariable
- PRTPOSTPROCESSINGSTAGEDESTROY rtPostProcessingStageDestroy
- PRTPOSTPROCESSINGSTAGEGETCONTEXT rtPostProcessingStageGetContext
- PRTPOSTPROCESSINGSTAGEQUERYVARIABLE rtPostProcessingStageQueryVariable
- PRTPOSTPROCESSINGSTAGEGETVARIABLECOUNT
rtPostProcessingStageGetVariableCount
- PRTPOSTPROCESSINGSTAGEGETVARIABLE rtPostProcessingStageGetVariable
- PRTPROGRAMCREATEFROMPTXFILE rtProgramCreateFromPTXFile
- PRTPROGRAMCREATEFROMPTXSTRING rtProgramCreateFromPTXString
- PRTPROGRAMDECLAREVARIABLE rtProgramDeclareVariable
- PRTPROGRAMDESTROY rtProgramDestroy
- PRTPROGRAMGETCONTEXT rtProgramGetContext
- PRTPROGRAMGETID rtProgramGetId
- PRTPROGRAMGETVARIABLE rtProgramGetVariable
- PRTPROGRAMGETVARIABLECOUNT rtProgramGetVariableCount
- PRTPROGRAMQUERYVARIABLE rtProgramQueryVariable
- PRTPROGRAMMOVEVARIABLE rtProgramRemoveVariable
- PRTPROGRAMVALIDATE rtProgramValidate
- PRTDEPRECATEDAPIFUNCTION rtRemoteDeviceCreate

- PRTDEPRECATEDAPIFUNCTION rtRemoteDeviceDestroy
- PRTDEPRECATEDAPIFUNCTION rtRemoteDeviceGetAttribute
- PRTDEPRECATEDAPIFUNCTION rtRemoteDeviceRelease
- PRTDEPRECATEDAPIFUNCTION rtRemoteDeviceReserve
- PRTSELECTORCREATE rtSelectorCreate
- PRTSELECTORDECLAREVARIABLE rtSelectorDeclareVariable
- PRTSELECTORDESTROY rtSelectorDestroy
- PRTSELECTORGETCHILD rtSelectorGetChild
- PRTSELECTORGETCHILDCOUNT rtSelectorGetChildCount
- PRTSELECTORGETCHILDTYPE rtSelectorGetChildType
- PRTSELECTORGETCONTEXT rtSelectorGetContext
- PRTSELECTORGETVARIABLE rtSelectorGetVariable
- PRTSELECTORGETVARIABLECOUNT rtSelectorGetVariableCount
- PRTSELECTORGETVISITPROGRAM rtSelectorGetVisitProgram
- PRTSELECTORQUERYVARIABLE rtSelectorQueryVariable
- PRTSELECTORREMOVEVARIABLE rtSelectorRemoveVariable
- PRTSELECTORSETCHILD rtSelectorSetChild
- PRTSELECTORSETCHILDCOUNT rtSelectorSetChildCount
- PRTSELECTORSETVISITPROGRAM rtSelectorSetVisitProgram
- PRTSELECTORVALIDATE rtSelectorValidate
- PRTTEXTURESAMPLERCREATE rtTextureSamplerCreate
- PRTTEXTURESAMPLERCREATEFROMGLIMAGE rtTextureSamplerCreateFromGLImage
- PRTTEXTURESAMPLERDESTROY rtTextureSamplerDestroy
- PRTTEXTURESAMPLERGLREGISTER rtTextureSamplerGLRegister
- PRTTEXTURESAMPLERGLUNREGISTER rtTextureSamplerGLUnregister
- PRTTEXTURESAMPLERGETARRAYSIZE rtTextureSamplerGetArraySize
- PRTTEXTURESAMPLERGETBUFFER rtTextureSamplerGetBuffer
- PRTTEXTURESAMPLERGETCONTEXT rtTextureSamplerGetContext
- PRTTEXTURESAMPLERGETFILTERINGMODES rtTextureSamplerGetFilteringModes
- PRTTEXTURESAMPLERGETGLIMAGEID rtTextureSamplerGetGLImageId
- PRTTEXTURESAMPLERGETID rtTextureSamplerGetId
- PRTTEXTURESAMPLERGETINDEXINGMODE rtTextureSamplerGetIndexingMode
- PRTTEXTURESAMPLERGETMAXANISOTROPY rtTextureSamplerGetMaxAnisotropy
- PRTTEXTURESAMPLERGETMIPLEVELCLAMP rtTextureSamplerGetMipLevelClamp
- PRTTEXTURESAMPLERGETMIPLEVELBIAS rtTextureSamplerGetMipLevelBias
- PRTTEXTURESAMPLERGETMIPLEVELCOUNT rtTextureSamplerGetMipLevelCount
- PRTTEXTURESAMPLERGETREADMODE rtTextureSamplerGetReadMode
- PRTTEXTURESAMPLERGETWRAPMODE rtTextureSamplerGetWrapMode
- PRTTEXTURESAMPLERSETARRAYSIZE rtTextureSamplerSetArraySize
- PRTTEXTURESAMPLERSETBUFFER rtTextureSamplerSetBuffer
- PRTTEXTURESAMPLERSETFILTERINGMODES rtTextureSamplerSetFilteringModes
- PRTTEXTURESAMPLERSETINDEXINGMODE rtTextureSamplerSetIndexingMode
- PRTTEXTURESAMPLERSETMAXANISOTROPY rtTextureSamplerSetMaxAnisotropy
- PRTTEXTURESAMPLERSETMIPLEVELCLAMP rtTextureSamplerSetMipLevelClamp
- PRTTEXTURESAMPLERSETMIPLEVELBIAS rtTextureSamplerSetMipLevelBias
- PRTTEXTURESAMPLERSETMIPLEVELCOUNT rtTextureSamplerSetMipLevelCount

- PRTTEXTURESAMPLERSETREADMODE rtTextureSamplerSetReadMode
- PRTTEXTURESAMPLERSETWRAPMODE rtTextureSamplerSetWrapMode
- PRTTEXTURESAMPLERVALIDATE rtTextureSamplerValidate
- PRTTRANSFORMCREATE rtTransformCreate
- PRTTRANSFORMDESTROY rtTransformDestroy
- PRTTRANSFORMGETCHILD rtTransformGetChild
- PRTTRANSFORMGETCHILDTYPE rtTransformGetChildType
- PRTTRANSFORMGETCONTEXT rtTransformGetContext
- PRTTRANSFORMGETMATRIX rtTransformGetMatrix
- PRTTRANSFORMSETCHILD rtTransformSetChild
- PRTTRANSFORMSETMATRIX rtTransformSetMatrix
- PRTTRANSFORMSETMOTIONRANGE rtTransformSetMotionRange
- PRTTRANSFORMGETMOTIONRANGE rtTransformGetMotionRange
- PRTTRANSFORMSETMOTIONBORDERMODE rtTransformSetMotionBorderMode
- PRTTRANSFORMGETMOTIONBORDERMODE rtTransformGetMotionBorderMode
- PRTTRANSFORMSETMOTIONKEYS rtTransformSetMotionKeys
- PRTTRANSFORMGETMOTIONKEYCOUNT rtTransformGetMotionKeyCount
- PRTTRANSFORMGETMOTIONKEYTYPE rtTransformGetMotionKeyType
- PRTTRANSFORMGETMOTIONKEYS rtTransformGetMotionKeys
- PRTTRANSFORMVALIDATE rtTransformValidate
- PRTVARIABLEGET1F rtVariableGet1f
- PRTVARIABLEGET1FV rtVariableGet1fv
- PRTVARIABLEGET1I rtVariableGet1i
- PRTVARIABLEGET1IV rtVariableGet1iv
- PRTVARIABLEGET1UI rtVariableGet1ui
- PRTVARIABLEGET1UIV rtVariableGet1uiv
- PRTVARIABLEGET1LL rtVariableGet1ll
- PRTVARIABLEGET1LLV rtVariableGet1llv
- PRTVARIABLEGET1ULL rtVariableGet1ull
- PRTVARIABLEGET1ULLV rtVariableGet1ullv
- PRTVARIABLEGET2F rtVariableGet2f
- PRTVARIABLEGET2FV rtVariableGet2fv
- PRTVARIABLEGET2I rtVariableGet2i
- PRTVARIABLEGET2IV rtVariableGet2iv
- PRTVARIABLEGET2UI rtVariableGet2ui
- PRTVARIABLEGET2UIV rtVariableGet2uiv
- PRTVARIABLEGET2LL rtVariableGet2ll
- PRTVARIABLEGET2LLV rtVariableGet2llv
- PRTVARIABLEGET2ULL rtVariableGet2ull
- PRTVARIABLEGET2ULLV rtVariableGet2ullv
- PRTVARIABLEGET3F rtVariableGet3f
- PRTVARIABLEGET3FV rtVariableGet3fv
- PRTVARIABLEGET3I rtVariableGet3i
- PRTVARIABLEGET3IV rtVariableGet3iv
- PRTVARIABLEGET3UI rtVariableGet3ui
- PRTVARIABLEGET3UIV rtVariableGet3uiv

- PRTVARIABLEGET3LL rtVariableGet3ll
- PRTVARIABLEGET3LLV rtVariableGet3llv
- PRTVARIABLEGET3ULL rtVariableGet3ull
- PRTVARIABLEGET3ULLV rtVariableGet3ullv
- PRTVARIABLEGET4F rtVariableGet4f
- PRTVARIABLEGET4FV rtVariableGet4fv
- PRTVARIABLEGET4I rtVariableGet4i
- PRTVARIABLEGET4IV rtVariableGet4iv
- PRTVARIABLEGET4UI rtVariableGet4ui
- PRTVARIABLEGET4UIV rtVariableGet4uiv
- PRTVARIABLEGET4LL rtVariableGet4ll
- PRTVARIABLEGET4LLV rtVariableGet4llv
- PRTVARIABLEGET4ULL rtVariableGet4ull
- PRTVARIABLEGET4ULLV rtVariableGet4ullv
- PRTVARIABLEGETANNOTATION rtVariableGetAnnotation
- PRTVARIABLEGETCONTEXT rtVariableGetContext
- PRTVARIABLEGETMATRIX2X2FV rtVariableGetMatrix2x2fv
- PRTVARIABLEGETMATRIX2X3FV rtVariableGetMatrix2x3fv
- PRTVARIABLEGETMATRIX2X4FV rtVariableGetMatrix2x4fv
- PRTVARIABLEGETMATRIX3X2FV rtVariableGetMatrix3x2fv
- PRTVARIABLEGETMATRIX3X3FV rtVariableGetMatrix3x3fv
- PRTVARIABLEGETMATRIX3X4FV rtVariableGetMatrix3x4fv
- PRTVARIABLEGETMATRIX4X2FV rtVariableGetMatrix4x2fv
- PRTVARIABLEGETMATRIX4X3FV rtVariableGetMatrix4x3fv
- PRTVARIABLEGETMATRIX4X4FV rtVariableGetMatrix4x4fv
- PRTVARIABLEGETNAME rtVariableGetName
- PRTVARIABLEGETOBJECT rtVariableGetObject
- PRTVARIABLEGETSIZE rtVariableGetSize
- PRTVARIABLEGETTYPE rtVariableGetType
- PRTVARIABLEGETUSERDATA rtVariableGetUserData
- PRTVARIABLESET1F rtVariableSet1f
- PRTVARIABLESET1FV rtVariableSet1fv
- PRTVARIABLESET1I rtVariableSet1i
- PRTVARIABLESET1IV rtVariableSet1iv
- PRTVARIABLESET1UI rtVariableSet1ui
- PRTVARIABLESET1UIV rtVariableSet1uiv
- PRTVARIABLESET1LL rtVariableSet1ll
- PRTVARIABLESET1LLV rtVariableSet1llv
- PRTVARIABLESET1ULL rtVariableSet1ull
- PRTVARIABLESET1ULLV rtVariableSet1ullv
- PRTVARIABLESET2F rtVariableSet2f
- PRTVARIABLESET2FV rtVariableSet2fv
- PRTVARIABLESET2I rtVariableSet2i
- PRTVARIABLESET2IV rtVariableSet2iv
- PRTVARIABLESET2UI rtVariableSet2ui
- PRTVARIABLESET2UIV rtVariableSet2uiv

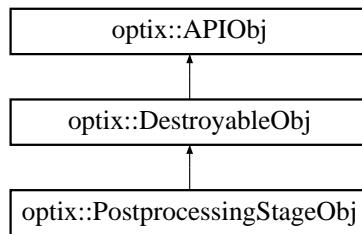
- [PRTVARIABLESET2LL rtVariableSet2ll](#)
- [PRTVARIABLESET2LLV rtVariableSet2llv](#)
- [PRTVARIABLESET2ULL rtVariableSet2ull](#)
- [PRTVARIABLESET2ULLV rtVariableSet2ullv](#)
- [PRTVARIABLESET3F rtVariableSet3f](#)
- [PRTVARIABLESET3FV rtVariableSet3fv](#)
- [PRTVARIABLESET3I rtVariableSet3i](#)
- [PRTVARIABLESET3IV rtVariableSet3iv](#)
- [PRTVARIABLESET3UI rtVariableSet3ui](#)
- [PRTVARIABLESET3UIV rtVariableSet3uiv](#)
- [PRTVARIABLESET3LL rtVariableSet3ll](#)
- [PRTVARIABLESET3LLV rtVariableSet3llv](#)
- [PRTVARIABLESET3ULL rtVariableSet3ull](#)
- [PRTVARIABLESET3ULLV rtVariableSet3ullv](#)
- [PRTVARIABLESET4F rtVariableSet4f](#)
- [PRTVARIABLESET4FV rtVariableSet4fv](#)
- [PRTVARIABLESET4I rtVariableSet4i](#)
- [PRTVARIABLESET4IV rtVariableSet4iv](#)
- [PRTVARIABLESET4UI rtVariableSet4ui](#)
- [PRTVARIABLESET4UIV rtVariableSet4uiv](#)
- [PRTVARIABLESET4LL rtVariableSet4ll](#)
- [PRTVARIABLESET4LLV rtVariableSet4llv](#)
- [PRTVARIABLESET4ULL rtVariableSet4ull](#)
- [PRTVARIABLESET4ULLV rtVariableSet4ullv](#)
- [PRTVARIABLESETMATRIX2X2FV rtVariableSetMatrix2x2fv](#)
- [PRTVARIABLESETMATRIX2X3FV rtVariableSetMatrix2x3fv](#)
- [PRTVARIABLESETMATRIX2X4FV rtVariableSetMatrix2x4fv](#)
- [PRTVARIABLESETMATRIX3X2FV rtVariableSetMatrix3x2fv](#)
- [PRTVARIABLESETMATRIX3X3FV rtVariableSetMatrix3x3fv](#)
- [PRTVARIABLESETMATRIX3X4FV rtVariableSetMatrix3x4fv](#)
- [PRTVARIABLESETMATRIX4X2FV rtVariableSetMatrix4x2fv](#)
- [PRTVARIABLESETMATRIX4X3FV rtVariableSetMatrix4x3fv](#)
- [PRTVARIABLESETMATRIX4X4FV rtVariableSetMatrix4x4fv](#)
- [PRTVARIABLESETOBJECT rtVariableSetObject](#)
- [PRTVARIABLESETUSERDATA rtVariableSetUserData](#)
- [PRTDEVICEGETWGLDEVICE rtDeviceGetWGLDevice](#)
- [PRTBUFFERCREATEFROMMD3D10RESOURCE rtBufferCreateFromD3D10Resource](#)
- [PRTBUFFERCREATEFROMMD3D11RESOURCE rtBufferCreateFromD3D11Resource](#)
- [PRTBUFFERCREATEFROMMD3D9RESOURCE rtBufferCreateFromD3D9Resource](#)
- [PRTBUFFERD3D10REGISTER rtBufferD3D10Register](#)
- [PRTBUFFERD3D10UNREGISTER rtBufferD3D10Unregister](#)
- [PRTBUFFERD3D11REGISTER rtBufferD3D11Register](#)
- [PRTBUFFERD3D11UNREGISTER rtBufferD3D11Unregister](#)
- [PRTBUFFERD3D9REGISTER rtBufferD3D9Register](#)
- [PRTBUFFERD3D9UNREGISTER rtBufferD3D9Unregister](#)
- [PRTBUFFERGETD3D10RESOURCE rtBufferGetD3D10Resource](#)

- PRTBUFFERGETD3D11RESOURCE rtBufferGetD3D11Resource
- PRTBUFFERGETD3D9RESOURCE rtBufferGetD3D9Resource
- PRTCXTSETD3D10DEVICE rtContextSetD3D10Device
- PRTCXTSETD3D11DEVICE rtContextSetD3D11Device
- PRTCXTSETD3D9DEVICE rtContextSetD3D9Device
- PRTDEVICEGETD3D9DEVICE rtDeviceGetD3D9Device
- PRTDEVICEGETD3D10DEVICE rtDeviceGetD3D10Device
- PRTDEVICEGETD3D11DEVICE rtDeviceGetD3D11Device
- PRTTEXTURESAMPLERCREATEFROMMD3D10RESOURCE
rtTextureSamplerCreateFromD3D10Resource
- PRTTEXTURESAMPLERCREATEFROMMD3D11RESOURCE
rtTextureSamplerCreateFromD3D11Resource
- PRTTEXTURESAMPLERCREATEFROMMD3D9RESOURCE
rtTextureSamplerCreateFromD3D9Resource
- PRTTEXTURESAMPLERD3D10REGISTER rtTextureSamplerD3D10Register
- PRTTEXTURESAMPLERD3D10UNREGISTER rtTextureSamplerD3D10Unregister
- PRTTEXTURESAMPLERD3D11REGISTER rtTextureSamplerD3D11Register
- PRTTEXTURESAMPLERD3D11UNREGISTER rtTextureSamplerD3D11Unregister
- PRTTEXTURESAMPLERD3D9REGISTER rtTextureSamplerD3D9Register
- PRTTEXTURESAMPLERD3D9UNREGISTER rtTextureSamplerD3D9Unregister
- PRTTEXTURESAMPLERGETD3D10RESOURCE rtTextureSamplerGetD3D10Resource
- PRTTEXTURESAMPLERGETD3D11RESOURCE rtTextureSamplerGetD3D11Resource
- PRTTEXTURESAMPLERGETD3D9RESOURCE rtTextureSamplerGetD3D9Resource
- PRTPROGRAMCREATEFROMPTXSTRINGS rtProgramCreateFromPTXStrings
- PRTPROGRAMCREATEFROMPTXFILES rtProgramCreateFromPTXFiles
- PRTGEOMETRYINSTANCEGETGEOMETRYTRIANGLES
rtGeometryInstanceGetGeometryTriangles
- PRTGEOMETRYINSTANCESETGEOMETRYTRIANGLES
rtGeometryInstanceSetGeometryTriangles
- PRTGEOMETRYTRIANGLESCREATE rtGeometryTrianglesCreate
- PRTGEOMETRYTRIANGLESDESTROY rtGeometryTrianglesDestroy
- PRTGEOMETRYTRIANGLESGETCONTEXT rtGeometryTrianglesGetContext
- PRTGEOMETRYTRIANGLESGETPRIMITIVEINDEXOFFSET
rtGeometryTrianglesGetPrimitiveIndexOffset
- PRTGEOMETRYTRIANGLESSETPRIMITIVEINDEXOFFSET
rtGeometryTrianglesSetPrimitiveIndexOffset
- PRTGEOMETRYTRIANGLESSETINDEXEDTRIANGLESDEPRECATED
rtGeometryTrianglesSetIndexedTrianglesDeprecated
- PRTGEOMETRYTRIANGLESSETTRIANGLESDEPRECATED
rtGeometryTrianglesSetTrianglesDeprecated
- PRTGEOMETRYTRIANGLESVALIDATE rtGeometryTrianglesValidate
- PRTLOG rtLog
- PRTCXTGETMAXCALLABLEPROGRAMDEPTH rtContextGetMaxCallableProgramDepth
- PRTCXTSETMAXCALLABLEPROGRAMDEPTH rtContextSetMaxCallableProgramDepth
- PRTCXTGETMAXTRACEDEPTH rtContextGetMaxTraceDepth
- PRTCXTSETMAXTRACEDEPTH rtContextSetMaxTraceDepth

- PRTPROGRAMCALLSITESETPOTENTIALCALLEES rtProgramCallsiteSetPotentialCallees
- PRTGETBUILDVERSION rtGetBuildVersion
- PRTOVERRIDESOTHERVERSION rtOverridesOtherVersion
- PRTGEOMETRYTRIANGLESSETATTRIBUTEPROGRAM
rtGeometryTrianglesSetAttributeProgram
- PRTGEOMETRYTRIANGLESGETATTRIBUTEPROGRAM
rtGeometryTrianglesGetAttributeProgram
- PRTGEOMETRYTRIANGLESDECLAREVARIABLE rtGeometryTrianglesDeclareVariable
- PRTGEOMETRYTRIANGLESQUERYVARIABLE rtGeometryTrianglesQueryVariable
- PRTGEOMETRYTRIANGLESREMOVEVARIABLE rtGeometryTrianglesRemoveVariable
- PRTGEOMETRYTRIANGLESGETVARIABLECOUNT rtGeometryTrianglesGetVariableCount
- PRTGEOMETRYTRIANGLESGETVARIABLE rtGeometryTrianglesGetVariable
- PRTGEOMETRYTRIANGLESSETPRETRANSFORMMATRIX
rtGeometryTrianglesSetPreTransformMatrix
- PRTGEOMETRYTRIANGLESGETPRETRANSFORMMATRIX
rtGeometryTrianglesGetPreTransformMatrix
- PRTGEOMETRYTRIANGLESSETPRIMITIVECOUNT rtGeometryTrianglesSetPrimitiveCount
- PRTGEOMETRYTRIANGLESGETPRIMITIVECOUNT rtGeometryTrianglesGetPrimitiveCount
- PRTGEOMETRYTRIANGLESSETTRIANGLESINDICES rtGeometryTrianglesSetTriangleIndices
- PRTGEOMETRYTRIANGLESSETVERTICES rtGeometryTrianglesSetVertices
- PRTGEOMETRYTRIANGLESSETMOTIONVERTICES rtGeometryTrianglesSetMotionVertices
- PRTGEOMETRYTRIANGLESSETMOTIONVERTEXESMULTIBUFFERDEPRECATED
rtGeometryTrianglesSetMotionVerticesMultiBufferDeprecated
- PRTGEOMETRYTRIANGLESSETMOTIONSTEPS rtGeometryTrianglesSetMotionSteps
- PRTGEOMETRYTRIANGLESGETMOTIONSTEPS rtGeometryTrianglesGetMotionSteps
- PRTGEOMETRYTRIANGLESSETMOTIONRANGE rtGeometryTrianglesSetMotionRange
- PRTGEOMETRYTRIANGLESGETMOTIONRANGE rtGeometryTrianglesGetMotionRange
- PRTGEOMETRYTRIANGLESSETMOTIONBORDERMODE
rtGeometryTrianglesSetMotionBorderMode
- PRTGEOMETRYTRIANGLESGETMOTIONBORDERMODE
rtGeometryTrianglesGetMotionBorderMode
- PRTGEOMETRYTRIANGLESSETBUILDFLAGS rtGeometryTrianglesSetBuildFlags
- PRTGEOMETRYTRIANGLESGETMATERIALCOUNT rtGeometryTrianglesGetMaterialCount
- PRTGEOMETRYTRIANGLESSETMATERIALCOUNT rtGeometryTrianglesSetMaterialCount
- PRTGEOMETRYTRIANGLESSETMATERIALINDICES rtGeometryTrianglesSetMaterialIndices
- PRTGEOMETRYTRIANGLESSETFLAGSPERMATERIAL
rtGeometryTrianglesSetFlagsPerMaterial
- PRTGEOMETRYTRIANGLESGETFLAGSPERMATERIAL
rtGeometryTrianglesGetFlagsPerMaterial
- PRTGROUPSETVISIBILITYMASK rtGroupSetVisibilityMask
- PRTGROUPGETVISIBILITYMASK rtGroupGetVisibilityMask
- PRTGEOMETRYGROUPSETFLAGS rtGeometryGroupSetFlags
- PRTGEOMETRYGROUPGETFLAGS rtGeometryGroupGetFlags
- PRTGEOMETRYGROUPSETVISIBILITYMASK rtGeometryGroupSetVisibilityMask
- PRTGEOMETRYGROUPGETVISIBILITYMASK rtGeometryGroupGetVisibilityMask
- PRTGEOMETRYSETFLAGS rtGeometrySetFlags
- PRTGEOMETRYGETFLAGS rtGeometryGetFlags
- PRTGEOMETRYTRIANGLESSETMOTIONVERTEXESMULTIBUFFER
rtGeometryTrianglesSetMotionVerticesMultiBuffer

7.34.1 Member Data Documentation

- 7.34.1.1 PRTACCELERATIONCREATE OptiXAPI::rtAccelerationCreate
- 7.34.1.2 PRTACCELERATIONDESTROY OptiXAPI::rtAccelerationDestroy
- 7.34.1.3 PRTACCELERATIONGETBUILDER OptiXAPI::rtAccelerationGetBuilder
- 7.34.1.4 PRTACCELERATIONGETCONTEXT OptiXAPI::rtAccelerationGetContext
- 7.34.1.5 PRTACCELERATIONGETDATA OptiXAPI::rtAccelerationGetData
- 7.34.1.6 PRTACCELERATIONGETDATASIZE OptiXAPI::rtAccelerationGetDataSize
- 7.34.1.7 PRTACCELERATIONGETPROPERTY OptiXAPI::rtAccelerationGetProperty
- 7.34.1.8 PRTACCELERATIONGETTRaverser OptiXAPI::rtAccelerationGetTraverser
- 7.34.1.9 PRTACCELERATIONISDIRTY OptiXAPI::rtAccelerationIsDirty
- 7.34.1.10 PRTACCELERATIONMARKDIRTY OptiXAPI::rtAccelerationMarkDirty
- 7.34.1.11 PRTACCELERATIONSETBUILDER OptiXAPI::rtAccelerationSetBuilder
- 7.34.1.12 PRTACCELERATIONSETDATA OptiXAPI::rtAccelerationSetData
- 7.34.1.13 PRTACCELERATIONSETPROPERTY OptiXAPI::rtAccelerationSetProperty
- 7.34.1.14 PRTACCELERATIONSETTRaverser OptiXAPI::rtAccelerationSetTraverser
- 7.34.1.15 PRTACCELERATIONVALIDATE OptiXAPI::rtAccelerationValidate
- 7.34.1.16 PRTBUFFERBINDPROGRESSIVESTREAM OptiXAPI::rtBufferBindProgressiveStream
- 7.34.1.17 PRTBUFFERCREATE OptiXAPI::rtBufferCreate
- 7.34.1.18 PRTBUFFERCREATEFORCUDA OptiXAPI::rtBufferCreateForCUDA
- 7.34.1.19 PRTBUFFERCREATEFROMD3D10RESOURCE OptiXAPI::rtBufferCreateFromD3D10Resource
- 7.34.1.20 PRTBUFFERCREATEFROMD3D11RESOURCE OptiXAPI::rtBufferCreateFromD3D11Resource
- 7.34.1.21 PRTBUFFERCREATEFROMD3D9RESOURCE OptiXAPI::rtBufferCreateFromD3D9Resource
- 7.34.1.22 PRTBUFFERCREATEFROMGLBO OptiXAPI::rtBufferCreateFromGLBO
- 7.34.1.23 PRTBUFFERD3D10REGISTER OptiXAPI::rtBufferD3D10Register
- 7.34.1.24 PRTBUFFERD3D10UNREGISTER OptiXAPI::rtBufferD3D10Unregister
- 7.34.1.25 PRTBUFFERD3D11REGISTER OptiXAPI::rtBufferD3D11Register



Public Member Functions

- void `destroy ()`
- void `validate ()`
- `Context getContext () const`
- `RTpostprocessingstage get ()`
- void `addReference ()`
- int `removeReference ()`
- virtual void `checkError (RTresult code) const`
- virtual void `checkError (RTresult code, Context context) const`
- void `checkErrorNoGetContext (RTresult code) const`

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

Static Public Member Functions

- static `Exception makeException (RTresult code, RTcontext context)`

7.35.1 Detailed Description

`PostProcessingStage` wraps the OptiX C API `RTpostprocessingstage` opaque type and its associated function set.

7.35.2 Member Function Documentation

7.35.2.1 void optix::APIObj::addReference () [inline], [inherited]

Increment the reference count for this object.

7.35.2.2 void optix::APIObj::checkError (

`RTresult code) const [inline], [virtual], [inherited]`

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

Reimplemented in `optix::ContextObj`.

7.35.2.3 void optix::APIObj::checkError (

RTresult *code*,

Context *context*) const [inline], [virtual], [inherited]

7.35.2.4 void optix::APIObj::checkErrorNoGetContext (

RTresult *code*) const [inline], [inherited]

7.35.2.5 Variable optix::PostprocessingStageObj::declareVariable (

const std::string & *name*) [inline]

7.35.2.6 void optix::PostprocessingStageObj::destroy() [inline], [virtual]

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

Implements optix::DestroyableObj.

7.35.2.7 RTpostprocessingstage optix::PostprocessingStageObj::get() [inline]

Get the underlying OptiX C API RTpostprocessingstage opaque pointer.

7.35.2.8 Context optix::PostprocessingStageObj::getContext() const [inline], [virtual]

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

Implements optix::APIObj.

7.35.2.9 Variable optix::PostprocessingStageObj::getVariable (

unsigned int *index*) const [inline]

7.35.2.10 unsigned int optix::PostprocessingStageObj::getVariableCount() const [inline]

7.35.2.11 Exception optix::APIObj::makeException (

RTresult *code*,

RTcontext *context*) [inline], [static], [inherited]

For backwards compatibility. Use [Exception::makeException](#) instead.

7.35.2.12 Variable optix::PostprocessingStageObj::queryVariable (

const std::string & *name*) const [inline]

7.35.2.13 int optix::APIObj::removeReference() [inline], [inherited]

Decrement the reference count for this object.

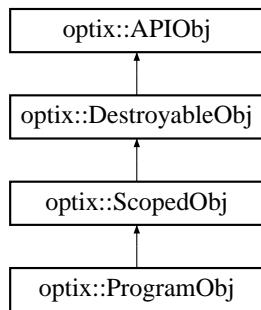
7.35.2.14 void optix::PostprocessingStageObj::validate() [inline], [virtual]

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

Implements optix::DestroyableObj.

7.36 optix::ProgramObj Class Reference

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](#)
- void [setCallsitePotentialCallees \(const std::string &callSiteName, const std::vector< int > &calleelids\)](#)
- RTprogram [get \(\)](#)
- void [addReference \(\)](#)
- int [removeReference \(\)](#)
- virtual void [checkError \(RTresult code\) const](#)
- virtual void [checkError \(RTresult code, Context context\) const](#)
- void [checkErrorNoGetContext \(RTresult code\) const](#)

- int [getId \(\) const](#)

Static Public Member Functions

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

7.36.1 Detailed Description

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

7.36.2 Member Function Documentation

7.36.2.1 void optix::APIObj::addReference() [inline], [inherited]

Increment the reference count for this object.

7.36.2.2 void optix::APIObj::checkError(

RTresult code) const [inline], [virtual], [inherited]

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

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

7.36.2.3 void optix::APIObj::checkError(

RTresult code,

Context context) const [inline], [virtual], [inherited]

7.36.2.4 void optix::APIObj::checkErrorNoGetContext(

RTresult code) const [inline], [inherited]

7.36.2.5 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](#).

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

call [rt\[ObjectType\]Destroy](#) on the underlying OptiX C object

Implements [optix::DestroyableObj](#).

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

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

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).

Implements [optix::APIObj](#).

7.36.2.9 int optix::ProgramObj::getId() const [inline]

Returns the device-side ID of this program object. See [rtProgramGetId](#)

7.36.2.10 Variable optix::ProgramObj::getVariable(

unsigned int index) const [inline], [virtual]

Query variable by index. See [rt\[ObjectType\]GetVariable](#).

Implements optix::ScopedObj.

7.36.2.11 **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.

7.36.2.12 **Exception optix::APIObj::makeException(**

RTresult code,

RTcontext context) [inline], [static], [inherited]

For backwards compatibility. Use [Exception::makeException](#) instead.

7.36.2.13 **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.

7.36.2.14 **int optix::APIObj::removeReference() [inline], [inherited]**

Decrement the reference count for this object.

7.36.2.15 **void optix::ProgramObj::removeVariable(**

Variable v) [inline], [virtual]

Remove a variable associated with this object.

Implements optix::ScopedObj.

7.36.2.16 **void optix::ProgramObj::setCallsitePotentialCallees(**

const std::string & callSiteName,

const std::vector< int > & calleeds) [inline]

7.36.2.17 **void optix::ProgramObj::validate() [inline], [virtual]**

call [rt\[ObjectType\]Validate](#) on the underlying OptiX C object

Implements optix::DestroyableObj.

7.37 optix::PtxEncryption Class Reference

Public Member Functions

- `PtxEncryption (RTcontext context, const char *publicVendorKey, const char *secretVendorKey)`
- `PtxEncryption (Context context, const char *publicVendorKey, const char *secretVendorKey)`
- `::std::string encrypt (const ::std::string &ptx) const`

Static Public Member Functions

- `::std::string getPrefix ()`

7.37.1 Constructor & Destructor Documentation

7.37.1.1 optix::PtxEncryption::PtxEncryption (
 RTcontext context,
 *const char * publicVendorKey,*
 *const char * secretVendorKey) [inline]*

Sets up the given context for encrypted PTX files.

This method uses the two keys to generate a session key that is used by the `encrypt()` method below. It also ensures that the same session key is computed inside OptiX without passing it over the API boundary. The session key is not deterministic since it depends on two random numbers chosen by both sides.

Afterwards, all PTX code passed to methods of this contexts must have been encrypted with the `encrypt()` method of this instance. There is no way to disable encryption again.

7.37.1.2 optix::PtxEncryption::PtxEncryption (
 Context context,
 *const char * publicVendorKey,*
 *const char * secretVendorKey) [inline]*

Sets up the given context for encrypted PTX files.

This method uses the two keys to generate a session key that is used by the `encrypt()` method below. It also ensures that the same session key is computed inside OptiX without passing it over the API boundary. The session key is not deterministic since it depends on two random numbers chosen by both sides.

Afterwards, all PTX code passed to methods of this contexts must have been encrypted with the `encrypt()` method of this instance. There is no way to disable encryption again.

7.37.2 Member Function Documentation

7.37.2.1 std::string optix::PtxEncryption::encrypt (
 const ::std::string & ptx) const [inline]

Encrypts the given PTX code using our session key.

7.37.2.2 ::std::string optix::PtxEncryption::getPrefix() [inline], [static]

The prefix used by the encrypted PTX code.

7.38 optix::Quaternion Class Reference

Public Member Functions

- [RT_HOSTDEVICE Quaternion \(\)](#)
- [RT_HOSTDEVICE Quaternion \(float x, float y, float z, float w\)](#)
- [RT_HOSTDEVICE Quaternion \(float4 v\)](#)
- [RT_HOSTDEVICE Quaternion \(const Quaternion &other\)](#)
- [RT_HOSTDEVICE Quaternion \(const float3 &axis, float angle\)](#)
- [RT_HOSTDEVICE void toMatrix \(float m\[16\]\) const](#)

Public Attributes

- [float4 m_q](#)

7.38.1 Detailed Description

Quaternion.

Description

Quaternion is a utility class for handling quaternions which are primarily useful for representing directions and rotations.

History

Quaternion was introduced in OptiX 5.0.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 OPTIXU_INLINE RT_HOSTDEVICE optix::Quaternion::Quaternion()

Construct identity quaternion.

7.38.2.2 OPTIXU_INLINE RT_HOSTDEVICE optix::Quaternion::Quaternion (

```
    float x,  
    float y,  
    float z,  
    float w )
```

Construct from coordinates x, y, z, w.

7.38.2.3 OPTIXU_INLINE RT_HOSTDEVICE optix::Quaternion::Quaternion (float4 v)

Construct from float4.

7.38.2.4 OPTIXU_INLINE RT_HOSTDEVICE optix::Quaternion::Quaternion (const Quaternion & other)

Copy constructor.

7.38.2.5 OPTIXU_INLINE RT_HOSTDEVICE optix::Quaternion::Quaternion (const float3 & axis, float angle)

Construct from axis and angle (in degrees)

7.38.3 Member Function Documentation

7.38.3.1 OPTIXU_INLINE RT_HOSTDEVICE void optix::Quaternion::toMatrix (float m[16]) const

From quaternion to rotation matrix.

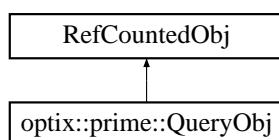
7.38.4 Member Data Documentation

7.38.4.1 float4 optix::Quaternion::m_q

quaternion x, y, z, w

7.39 optix::prime::QueryObj Class Reference

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 ()`

7.39.1 Detailed Description

Encapsulates an OptiX Prime query.

The purpose of a query is to coordinate the intersection of rays with a model.

7.39.2 Member Function Documentation

7.39.2.1 void optix::prime::QueryObj::execute (*unsigned hint*) [inline]

Executes a raytracing query. See [rtpQueryExecute](#).

7.39.2.2 void optix::prime::QueryObj::finish () [inline]

Blocks current thread until query is finished. See [rtpQueryFinish](#).

7.39.2.3 Context optix::prime::QueryObj::getContext () [inline]

Returns the context associated within this object.

7.39.2.4 RTPquery optix::prime::QueryObj::getRTPquery () [inline]

Returns the `RTPquery` query stored within this object.

7.39.2.5 int optix::prime::QueryObj::isFinished () [inline]

Polls the status of a query. See [rtpQueryGetFinished](#).

7.39.2.6 void optix::prime::QueryObj::setCudaStream (*cudaStream_t stream*) [inline]

Sets a stream for a query. See [rtpQuerySetCudaStream](#).

7.39.2.7 void optix::prime::QueryObj::setHits (*RTPsize count*, *RTPbufferformat format*, *RTPbuffertype type*, *void * hits*) [inline]

Sets a hit buffer for the query. See [rtpQuerySetHits](#).

7.39.2.8 void optix::prime::QueryObj::setHits (const BufferDesc & *hits*) [inline]

Sets a hit buffer for the query from a buffer description. See [rtpQuerySetHits](#).

7.39.2.9 void optix::prime::QueryObj::setRays (RTPsize *count*, RTPbufferformat *format*, RTPbuffertype *type*, void * *rays*) [inline]

Creates a buffer descriptor and sets the rays of a query. See [rtpQuerySetRays](#).

7.39.2.10 void optix::prime::QueryObj::setRays (const BufferDesc & *rays*) [inline]

Sets the rays of a query from a buffer descriptor. See [rtpQuerySetRays](#).

7.40 Ray Struct Reference

Public Attributes

- float3 *origin*
- float3 *direction*
- unsigned int *ray_type*
- float *tmin*
- float *tmax*

7.40.1 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](#)

7.40.2 Member Data Documentation

7.40.2.1 float3 Ray::direction

The direction of the ray.

7.40.2.2 float3 Ray::origin

The origin of the ray.

7.40.2.3 `unsigned int Ray::ray_type`

The ray type associated with this ray.

7.40.2.4 `float Ray::tmax`

The max extent associated with this ray.

7.40.2.5 `float Ray::tmin`

The min extent associated with this ray.

7.41 rtCallableProgramSizeofWrapper< T > Struct Template Reference

Static Public Attributes

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

7.41.1 Member Data Documentation

7.41.1.1 template<typename T> const size_t rtCallableProgramSizeofWrapper< T >::`value` = `sizeof(T)` [static]

7.42 rtCallableProgramSizeofWrapper< void > Struct Template Reference

Static Public Attributes

- static const size_t `value` = 0

7.42.1 Member Data Documentation

7.42.1.1 const size_t rtCallableProgramSizeofWrapper< void >::`value` = 0 [static]

7.43 rti_internal_typeinfo::rti_typeenum< T > Struct Template Reference

Static Public Attributes

- static const int `m_typeenum` = `_OPTIX_TYPE_ENUM_UNKNOWN`

7.43.1 Member Data Documentation

7.43.1.1 `template<typename T> const int rti_internal_typeinfo::rti_typeenum< T >::m_typeenum = _OPTIX_TYPE_ENUM_UNKNOWN [static]`

7.44 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`

7.44.1 Member Data Documentation

7.44.1.1 `template<typename T> const int rti_internal_typeinfo::rti_typeenum< optix::boundCallableProgramId< T > >::m_typeenum = _OPTIX_TYPE_ENUM_PROGRAM_AS_ID [static]`

7.45 rti_internal_typeinfo::rti_typeenum< optix::callableProgramId< T > > Struct Template Reference

Static Public Attributes

- `static const int m_typeenum = _OPTIX_TYPE_ENUM_PROGRAM_ID`

7.45.1 Member Data Documentation

7.45.1.1 `template<typename T> const int rti_internal_typeinfo::rti_typeenum< optix::callableProgramId< T > >::m_typeenum = _OPTIX_TYPE_ENUM_PROGRAM_ID [static]`

7.46 rti_internal_typeinfo::rti_typeinfo Struct Reference

Public Attributes

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

7.46.1 Member Data Documentation

7.46.1.1 `unsigned int rti_internal_typeinfo::rti_typeinfo::kind`

7.46.1.2 `unsigned int rti_internal_typeinfo::rti_typeinfo::size`

7.47 rtObject Struct Reference

Protected Member Functions

- `void never_call ()`

Protected Attributes

- `unsigned int handle`

7.47.1 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`

7.47.2 Member Function Documentation

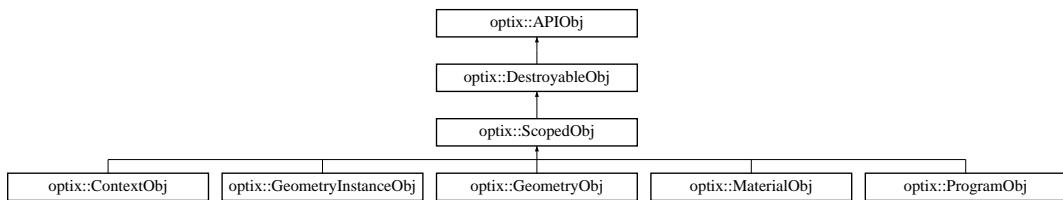
7.47.2.1 `void rtObject::never_call () [inline], [protected]`

7.47.3 Member Data Documentation

7.47.3.1 `unsigned int rtObject::handle [protected]`

7.48 optix::ScopedObj Class Reference

Inheritance diagram for optix::ScopedObj:



Public Member Functions

- virtual ~`ScopedObj` ()
- virtual `Variable declareVariable (const std::string &name)=0`
- virtual `Variable queryVariable (const std::string &name) const =0`
- virtual void `removeVariable (Variable v)=0`
- virtual unsigned int `getVariableCount () const =0`
- virtual `Variable getVariable (unsigned int index) const =0`
- virtual void `destroy ()=0`
- virtual void `validate ()=0`
- void `addReference ()`
- int `removeReference ()`
- virtual `Context getContext () const =0`
- virtual void `checkError (RTresult code) const`
- virtual void `checkError (RTresult code, Context context) const`
- void `checkErrorNoGetContext (RTresult code) const`

Static Public Member Functions

- static `Exception makeException (RTresult code, RTcontext context)`

7.48.1 Detailed Description

Base class for all objects which are OptiX variable containers.

Wraps:

- RTcontext
- RTgeometry
- RTgeometryinstance
- RTmaterial
- RTprogram

7.48.2 Constructor & Destructor Documentation

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

7.48.3 Member Function Documentation

7.48.3.1 void optix::APIObj::addReference() [inline], [inherited]

Increment the reference count for this object.

7.48.3.2 void optix::APIObj::checkError(

RTResult *code*) const [inline], [virtual], [inherited]

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

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

7.48.3.3 void optix::APIObj::checkError(

RTResult *code*,

Context *context*) const [inline], [virtual], [inherited]

7.48.3.4 void optix::APIObj::checkErrorNoGetContext(

RTResult *code*) const [inline], [inherited]

7.48.3.5 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](#).

7.48.3.6 virtual void optix::DestroyableObj::destroy() [pure virtual], [inherited]

call [rt\[ObjectType\]Destroy](#) on the underlying OptiX C object

Implemented in [optix::CommandListObj](#), [optix::PostprocessingStageObj](#), [optix::BufferObj](#), [optix::TextureSamplerObj](#), [optix::MaterialObj](#), [optix::GeometryTrianglesObj](#), [optix::GeometryObj](#), [optix::GeometryInstanceObj](#), [optix::AccelerationObj](#), [optix::SelectorObj](#), [optix::TransformObj](#), [optix::GeometryGroupObj](#), [optix::GroupObj](#), [optix::ProgramObj](#), and [optix::ContextObj](#).

7.48.3.7 virtual Context optix::APIObj::getContext() const [pure virtual], [inherited]

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).

Implemented in [optix::CommandListObj](#), [optix::PostprocessingStageObj](#), [optix::BufferObj](#), [optix::TextureSamplerObj](#), [optix::MaterialObj](#), [optix::GeometryTrianglesObj](#), [optix::GeometryObj](#), [optix::GeometryInstanceObj](#), [optix::AccelerationObj](#), [optix::SelectorObj](#), [optix::TransformObj](#), [optix::GeometryGroupObj](#), [optix::GroupObj](#), [optix::ProgramObj](#), [optix::ContextObj](#), and

optix::VariableObj.

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

Query variable by index. See [rt\[ObjectType\]GetVariable](#).

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

7.48.3.9 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](#).

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

For backwards compatibility. Use [Exception::makeException](#) instead.

7.48.3.11 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](#).

7.48.3.12 int optix::APIObj::removeReference () [inline], [inherited]

Decrement the reference count for this object.

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

Remove a variable associated with this object.

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

7.48.3.14 virtual void optix::DestroyableObj::validate () [pure virtual], [inherited]

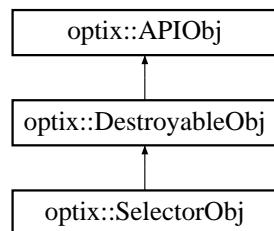
call [rt\[ObjectType\]Validate](#) on the underlying OptiX C object

Implemented in [optix::CommandListObj](#), [optix::PostprocessingStageObj](#), [optix::BufferObj](#),

`optix::TextureSamplerObj`, `optix::MaterialObj`, `optix::GeometryTrianglesObj`, `optix::GeometryObj`, `optix::GeometryInstanceObj`, `optix::AccelerationObj`, `optix::SelectorObj`, `optix::TransformObj`, `optix::GeometryGroupObj`, `optix::GroupObj`, `optix::ProgramObj`, and `optix::ContextObj`.

7.49 optix::SelectorObj Class Reference

Inheritance diagram for `optix::SelectorObj`:



Public Member Functions

- `void destroy ()`
- `void validate ()`
- `Context getContext () const`
- `RTselector get ()`
- `void addReference ()`
- `int removeReference ()`
- `virtual void checkError (RTresult code) const`
- `virtual void checkError (RTresult code, Context context) const`
- `void checkErrorNoGetContext (RTresult code) const`

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

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

Static Public Member Functions

- static Exception makeException (RTresult code, RTcontext context)

7.49.1 Detailed Description

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

7.49.2 Member Function Documentation

7.49.2.1 template<typename T> unsigned int optix::SelectorObj::addChild (T *child*) [inline]

Set a new child in this group and returns its new index. See [rtSelectorSetChild](#).

7.49.2.2 void optix::APIObj::addReference () [inline], [inherited]

Increment the reference count for this object.

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

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

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

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

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

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

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

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

Implements [optix::DestroyableObj](#).

7.49.2.8 RTselector optix::SelectorObj::get() [inline]

Get the underlying OptiX C API RTselector opaque pointer.

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

Query an indexed child within this group. See [rtSelectorGetChild](#).

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

Query the number of children for this group. See [rtSelectorGetChildCount](#).

7.49.2.11 template<typename T > unsigned int optix::SelectorObj::getChildIndex(T child) const [inline]

Query a child in this group for its index. See [rtSelectorGetChild](#).

7.49.2.12 RTobjecttype optix::SelectorObj::getChildType(unsigned int index) const [inline]

Query indexed child's type. See [rtSelectorGetChildType](#).

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

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).

Implements [optix::APIObj](#).

7.49.2.14 Variable optix::SelectorObj::getVariable(unsigned int index) const [inline]**7.49.2.15 unsigned int optix::SelectorObj::getVariableCount() const [inline]****7.49.2.16 Program optix::SelectorObj::getVisitProgram() const [inline]**

Get the visitor program for this selector. See [rtSelectorGetVisitProgram](#).

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

For backwards compatibility. Use [Exception::makeException](#) instead.

7.49.2.18 Variable optix::SelectorObj::queryVariable(const std::string & name) const [inline]**7.49.2.19 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.

7.49.2.20 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.

7.49.2.21 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.

7.49.2.22 int optix::APIObj::removeReference () [inline], [inherited]

Decrement the reference count for this object.

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

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

Set an indexed child child of this group. See [rtSelectorSetChild](#).

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

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

7.49.2.26 void optix::SelectorObj::setVisitProgram (
Program *program*) [inline]

Set the visitor program for this selector. See [rtSelectorSetVisitProgram](#)

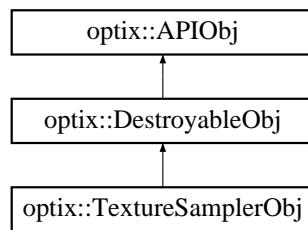
7.49.2.27 void optix::SelectorObj::validate () [inline], [virtual]

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

Implements [optix::DestroyableObj](#).

7.50 optix::TextureSamplerObj Class Reference

Inheritance diagram for optix::TextureSamplerObj:



Public Member Functions

- void `destroy ()`
- void `validate ()`
- `Context getContext () const`
- `RTtexturesampler get ()`
- void `addReference ()`
- int `removeReference ()`
- virtual void `checkError (RTresult code) const`
- virtual void `checkError (RTresult code, Context context) const`
- void `checkErrorNoGetContext (RTresult code) const`

- void `setMipLevelCount (unsigned int num_mip_levels)`
- unsigned int `getMipLevelCount () const`
- void `setArraySize (unsigned int num_textures_in_array)`
- unsigned int `getArraySize () const`
- void `setWrapMode (unsigned int dim, RTwrapmode wrapmode)`
- `RTwrapmode getWrapMode (unsigned int dim) const`
- void `setFilteringModes (RTfiltermode minification, RTfiltermode magnification, RTfiltermode mipmapping)`
- void `getFilteringModes (RTfiltermode &minification, RTfiltermode &magnification, RTfiltermode &mipmapping) const`
- void `setMaxAnisotropy (float value)`
- float `getMaxAnisotropy () const`
- void `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 ()

Static Public Member Functions

- static Exception makeException (RTresult code, RTcontext context)

7.50.1 Detailed Description

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

7.50.2 Member Function Documentation

7.50.2.1 void optix::APIObj::addReference () [inline], [inherited]

Increment the reference count for this object.

7.50.2.2 void optix::APIObj::checkError (

RTresult *code*) const [inline], [virtual], [inherited]

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

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

7.50.2.3 void optix::APIObj::checkError (

RTresult *code*,

Context *context*) const [inline], [virtual], [inherited]

7.50.2.4 void optix::APIObj::checkErrorNoGetContext (

RTresult *code*) const [inline], [inherited]

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

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

Implements [optix::DestroyableObj](#).

7.50.2.6 RTtexturesampler optix::TextureSamplerObj::get () [inline]

Get the underlying OptiX C API RTtexturesampler opaque pointer.

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

Deprecated in OptiX 4.0 Query the texture array size for this sampler. See

[rtTextureSamplerGetArraySize](#)

```
7.50.2.8 Buffer optix::TextureSamplerObj::getBuffer (
    unsigned int texture_array_idx,
    unsigned int mip_level ) const [inline]
```

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

7.50.2.9 Buffer optix::TextureSamplerObj::getBuffer () const [inline]

Get the underlying buffer used for texture storage. See [rtTextureSamplerGetBuffer](#).

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

Retrieve the context this object is associated with. See [rt\[ObjectType\]GetContext](#).

Implements [optix::APIObj](#).

```
7.50.2.11 void optix::TextureSamplerObj::getFilteringModes (
    RTfiltermode & minification,
    RTfiltermode & magnification,
    RTfiltermode & mipmapping ) const [inline]
```

Query filtering modes for this sampler. See [rtTextureSamplerGetFilteringModes](#).

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

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

7.50.2.13 RTtextureindexmode optix::TextureSamplerObj::getIndexingMode () const [inline]

Query texture indexing mode for this sampler. See [rtTextureSamplerGetIndexingMode](#).

7.50.2.14 float optix::TextureSamplerObj::getMaxAnisotropy () const [inline]

Query maximum anisotropy for this sampler. See [rtTextureSamplerGetMaxAnisotropy](#).

7.50.2.15 float optix::TextureSamplerObj::getMipLevelBias () const [inline]

Query mipmap offset for this sampler. See [rtTextureSamplerGetMipLevelBias](#).

```
7.50.2.16 void optix::TextureSamplerObj::getMipLevelClamp (
    float & minLevel,
    float & maxLevel ) const [inline]
```

Query minimum and maximum mipmap levels for this sampler. See [rtTextureSamplerGetMipLevelClamp](#).

7.50.2.17 `unsigned int optix::TextureSamplerObj::getMipLevelCount() const [inline]`

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

7.50.2.18 `RTtexturereadmode optix::TextureSamplerObj::getReadMode() const [inline]`

Query texture read mode for this sampler. See [rtTextureSamplerGetReadMode](#).

7.50.2.19 `RTwrapmode optix::TextureSamplerObj::getWrapMode(unsigned int dim) const [inline]`

Query the texture wrap mode for this sampler. See [rtTextureSamplerGetWrapMode](#).

7.50.2.20 `Exception optix::APIObj::makeException(RTResult code, RTcontext context) [inline], [static], [inherited]`

For backwards compatibility. Use [Exception::makeException](#) instead.

7.50.2.21 `void optix::TextureSamplerObj::registerGLTexture() [inline]`

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

7.50.2.22 `int optix::APIObj::removeReference() [inline], [inherited]`

Decrement the reference count for this object.

7.50.2.23 `void optix::TextureSamplerObj::setArraySize(unsigned int num_textures_in_array) [inline]`

Deprecated in OptiX 4.0 Set the texture array size for this sampler. See [rtTextureSamplerSetArraySize](#)

7.50.2.24 `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](#).

7.50.2.25 `void optix::TextureSamplerObj::setBuffer(Buffer buffer) [inline]`

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

7.50.2.26 `void optix::TextureSamplerObj::setFilteringModes(RTfiltermode minification, RTfiltermode magnification,`

RTfiltermode *mipmapping*) [inline]

Set filtering modes for this sampler. See [rtTextureSamplerSetFilteringModes](#).

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

Set texture indexing mode for this sampler. See [rtTextureSamplerSetIndexingMode](#).

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

Set maximum anisotropy for this sampler. See [rtTextureSamplerSetMaxAnisotropy](#).

7.50.2.29 void optix::TextureSamplerObj::setMipLevelBias (float *value*) [inline]

Set mipmap offset for this sampler. See [rtTextureSamplerSetMipLevelBias](#).

7.50.2.30 void optix::TextureSamplerObj::setMipLevelClamp (float *minLevel*, float *maxLevel*) [inline]

Set minimum and maximum mipmap levels for this sampler. See [rtTextureSamplerSetMipLevelClamp](#).

7.50.2.31 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](#).

7.50.2.32 void optix::TextureSamplerObj::setReadMode (RTtexureremode *readmode*) [inline]

Set texture read mode for this sampler. See [rtTextureSamplerSetReadMode](#).

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

Set the texture wrap mode for this sampler. See [rtTextureSamplerSetWrapMode](#).

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

Declare the texture's buffer as mutable and inaccessible by OptiX. See [rtTextureSamplerGLUnregister](#).

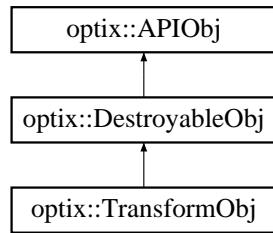
7.50.2.35 void optix::TextureSamplerObj::validate () [inline], [virtual]

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

Implements `optix::DestroyableObj`.

7.51 optix::TransformObj Class Reference

Inheritance diagram for optix::TransformObj:



Public Member Functions

- void `destroy ()`
- void `validate ()`
- `Context getContext () const`
- `RTtransform get ()`
- void `addReference ()`
- int `removeReference ()`
- virtual void `checkError (RTresult code) const`
- virtual void `checkError (RTresult code, Context context) const`
- void `checkErrorNoGetContext (RTresult code) const`

- template<typename T >
void `setChild (T child)`
- template<typename T >
`T getChild () const`
- `RTobjecttype getChildType () const`

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

- void `setMotionRange (float timeBegin, float timeEnd)`
- void `getMotionRange (float &timeBegin, float &timeEnd)`
- void `setMotionBorderMode (RTmotionbordermode beginMode, RTmotionbordermode endMode)`
- void `getMotionBorderMode (RTmotionbordermode &beginMode, RTmotionbordermode &endMode)`
- void `setMotionKeys (unsigned int n, RTmotionkeytype type, const float *keys)`
- unsigned int `getMotionKeyCount ()`
- `RTmotionkeytype getMotionKeyType ()`
- void `getMotionKeys (float *keys)`

Static Public Member Functions

- static `Exception makeException (RTresult code, RTcontext context)`

7.51.1 Detailed Description

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

7.51.2 Member Function Documentation

7.51.2.1 void optix::APIObj::addReference() [inline], [inherited]

Increment the reference count for this object.

7.51.2.2 void optix::APIObj::checkError(

RTresult code) const [inline], [virtual], [inherited]

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

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

7.51.2.3 void optix::APIObj::checkError(

RTresult code,

Context context) const [inline], [virtual], [inherited]

7.51.2.4 void optix::APIObj::checkErrorNoGetContext(

RTresult code) const [inline], [inherited]

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

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

Implements [optix::DestroyableObj](#).

7.51.2.6 RTtransform optix::TransformObj::get() [inline]

Get the underlying OptiX C API RTtransform opaque pointer.

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

Set the child node of this transform. See `rtTransformGetChild`.

7.51.2.8 RTobjecttype optix::TransformObj::getChildType() const [inline]

Query child's type. See `rtTransformGetChildType`.

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

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

Implements [optix::APIObj](#).

7.51.2.10 void optix::TransformObj::getMatrix(

bool transpose,

float * matrix,

```
float * inverse_matrix ) const [inline]
```

Get the transform matrix for this node. See [rtTransformGetMatrix](#).

7.51.2.11 void optix::TransformObj::getMotionBorderMode (
 RTmotionbordermode & *beginMode*,
 RTmotionbordermode & *endMode*) [inline]

Query the motion border mode for this transform. See [rtTransformGetMotionBorderMode](#).

7.51.2.12 unsigned int optix::TransformObj::getMotionKeyCount() [inline]

Query the number of motion keys for this transform. See [rtTransformGetMotionKeyCount](#).

7.51.2.13 void optix::TransformObj::getMotionKeys (
 float * *keys*) [inline]

Query the motion keys for this transform. See [rtTransformGetMotionKeys](#).

7.51.2.14 RTmotionkeytype optix::TransformObj::getMotionKeyType() [inline]

Query the motion key type for this transform. See [rtTransformGetMotionKeyType](#).

7.51.2.15 void optix::TransformObj::getMotionRange (
 float & *timeBegin*,
 float & *timeEnd*) [inline]

Query the motion time range for this transform. See [rtTransformGetMotionRange](#).

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

For backwards compatibility. Use [Exception::makeException](#) instead.

7.51.2.17 int optix::APIObj::removeReference() [inline], [inherited]

Decrement the reference count for this object.

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

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

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

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

7.51.2.20 void optix::TransformObj::setMotionBorderMode (
RTmotionbordermode *beginMode*,
RTmotionbordermode *endMode*) [inline]

Set the motion border mode for this transform. See [rtTransformSetMotionBorderMode](#).

7.51.2.21 void optix::TransformObj::setMotionKeys (
unsigned int *n*,
RTmotionkeytype *type*,
const float * *keys*) [inline]

Set the motion keys for this transform. See [rtTransformSetMotionKeys](#).

7.51.2.22 void optix::TransformObj::setMotionRange (
float *timeBegin*,
float *timeEnd*) [inline]

Set the motion time range for this transform. See [rtTransformSetMotionRange](#).

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

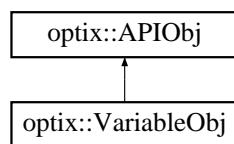
call [rt\[ObjectType\]Validate](#) on the underlying OptiX C object

Implements [optix::DestroyableObj](#).

7.52 optix::buffer< T, Dim >::type< T2 > Struct Template Reference

7.53 optix::VariableObj Class Reference

Inheritance diagram for [optix::VariableObj](#):



Public Member Functions

- [Context getContext \(\) const](#)
- [std::string getName \(\) const](#)
- [std::string getAnnotation \(\) const](#)
- [RTobjecttype getType \(\) const](#)
- [RTvariable get \(\)](#)
- [RTsize getSize \(\) const](#)
- [void addReference \(\)](#)
- [int removeReference \(\)](#)
- [virtual void checkError \(RTresult code\) const](#)

- virtual void `checkError (RTresult code, Context context) const`
- void `checkErrorNoGetContext (RTresult code) 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)`

Long Long setters

Set variable to have a long long value.

- void `setLongLong (long long i1)`

- void `setLongLong` (long long i1, long long i2)
- void `setLongLong` (long long i1, long long i2, long long i3)
- void `setLongLong` (long long i1, long long i2, long long i3, long long i4)
- void `setLongLong` (optix::longlong2 i)
- void `setLongLong` (optix::longlong3 i)
- void `setLongLong` (optix::longlong4 i)
- void `set1llv` (const long long *i)
- void `set2llv` (const long long *i)
- void `set3llv` (const long long *i)
- void `set4llv` (const long long *i)

Unsigned Long Long setters

Set variable to have an unsigned long long value.

- void `setULongLong` (unsigned long long u1)
- void `setULongLong` (unsigned long long u1, unsigned long long u2)
- void `setULongLong` (unsigned long long u1, unsigned long long u2, unsigned long long u3)
- void `setULongLong` (unsigned long long u1, unsigned long long u2, unsigned long long u3, unsigned long long u4)
- void `setULongLong` (optix::ulonglong2 u)
- void `setULongLong` (optix::ulonglong3 u)
- void `setULongLong` (optix::ulonglong4 u)
- void `set1ullv` (const unsigned long long *u)
- void `set2ullv` (const unsigned long long *u)
- void `set3ullv` (const unsigned long long *u)
- void `set4ullv` (const unsigned long long *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](#)
- unsigned long long [getULongLong \(\) const](#)
- optix::ulonglong2 [getULongLong2 \(\) const](#)
- optix::ulonglong3 [getULongLong3 \(\) const](#)
- optix::ulonglong4 [getULongLong4 \(\) const](#)
- void [getULongLong \(unsigned long long &ull1\) const](#)
- void [getULongLong \(unsigned long long &ull1, unsigned long long &ull2\) const](#)
- void [getULongLong \(unsigned long long &ull1, unsigned long long &ull2, unsigned long long &ull3\) const](#)
- void [getULongLong \(unsigned long long &ull1, unsigned long long &ull2, unsigned long long &ull3, unsigned long long &ull4\) const](#)
- long long [getLongLong \(\) const](#)
- optix::longlong2 [getLongLong2 \(\) const](#)
- optix::longlong3 [getLongLong3 \(\) const](#)
- optix::longlong4 [getLongLong4 \(\) const](#)
- void [getLongLong \(long long &ll1\) const](#)
- void [getLongLong \(long long &ll1, long long &ll2\) const](#)
- void [getLongLong \(long long &ll1, long long &ll2, long long &ll3\) const](#)
- void [getLongLong \(long long &ll1, long long &ll2, long long &ll3, long long &ll4\) 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](#)

Static Public Member Functions

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

7.53.1 Detailed Description

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

See the OptiX Programming Guide for a 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.

7.53.2 Member Function Documentation

7.53.2.1 void [optix::APIObj::addReference \(\) \[inline\], \[inherited\]](#)

Increment the reference count for this object.

7.53.2.2 void [optix::APIObj::checkError \(RTResult code \) const \[inline\], \[virtual\], \[inherited\]](#)

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

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

7.53.2.3 void optix::APIObj::checkError (
 RTresult code,
 Context context) const [inline], [virtual], [inherited]

7.53.2.4 void optix::APIObj::checkErrorNoGetContext (
 RTresult code) const [inline], [inherited]

7.53.2.5 RTvariable optix::VariableObj::get() [inline]

Get the OptiX C API object wrapped by this instance.

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

Retrieve the annotation associated with the variable.

7.53.2.7 Buffer optix::VariableObj::getBuffer() const [inline]

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

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

Implements [optix::APIObj](#).

7.53.2.9 float optix::VariableObj::getFloat() const [inline]

7.53.2.10 void optix::VariableObj::getFloat (
 float & f1) const [inline]

7.53.2.11 void optix::VariableObj::getFloat (
 float & f1,
 float & f2) const [inline]

7.53.2.12 void optix::VariableObj::getFloat (
 float & f1,
 float & f2,
 float & f3) const [inline]

7.53.2.13 void optix::VariableObj::getFloat (
 float & f1,
 float & f2,
 float & f3,

```
    float & f4 ) const [inline]
```

7.53.2.14 `optix::float2 optix::VariableObj::getFloat2() const [inline]`

7.53.2.15 `optix::float3 optix::VariableObj::getFloat3() const [inline]`

7.53.2.16 `optix::float4 optix::VariableObj::getFloat4() const [inline]`

7.53.2.17 `optix::GeometryGroup optix::VariableObj::getGeometryGroup() const [inline]`

7.53.2.18 `optix::GeometryInstance optix::VariableObj::getGeometryInstance() const [inline]`

7.53.2.19 `optix::Group optix::VariableObj::getGroup() const [inline]`

7.53.2.20 `int optix::VariableObj::getInt() const [inline]`

7.53.2.21 `void optix::VariableObj::getInt(`
 `int & i1) const [inline]`

7.53.2.22 `void optix::VariableObj::getInt(`
 `int & i1,`
 `int & i2) const [inline]`

7.53.2.23 `void optix::VariableObj::getInt(`
 `int & i1,`
 `int & i2,`
 `int & i3) const [inline]`

7.53.2.24 `void optix::VariableObj::getInt(`
 `int & i1,`
 `int & i2,`
 `int & i3,`
 `int & i4) const [inline]`

7.53.2.25 `optix::int2 optix::VariableObj::getInt2() const [inline]`

7.53.2.26 `optix::int3 optix::VariableObj::getInt3() const [inline]`

7.53.2.27 `optix::int4 optix::VariableObj::getInt4() const [inline]`

7.53.2.28 `long long optix::VariableObj::getLongLong() const [inline]`

7.53.2.29 `void optix::VariableObj::getLongLong(`
 `long long & lI1) const [inline]`

7.53.2.30 `void optix::VariableObj::getLongLong(`
 `long long & lI1,`

```
long long & l12 ) const [inline]

7.53.2.31 void optix::VariableObj::getLongLong (
    long long & l1,
    long long & l2,
    long long & l3 ) const [inline]

7.53.2.32 void optix::VariableObj::getLongLong (
    long long & l1,
    long long & l2,
    long long & l3,
    long long & l4 ) const [inline]

7.53.2.33 optix::longlong2 optix::VariableObj::getLongLong2( ) const [inline]

7.53.2.34 optix::longlong3 optix::VariableObj::getLongLong3( ) const [inline]

7.53.2.35 optix::longlong4 optix::VariableObj::getLongLong4( ) const [inline]

7.53.2.36 void optix::VariableObj::getMatrix2x2 (
    bool transpose,
    float * m ) const [inline]

7.53.2.37 void optix::VariableObj::getMatrix2x3 (
    bool transpose,
    float * m ) const [inline]

7.53.2.38 void optix::VariableObj::getMatrix2x4 (
    bool transpose,
    float * m ) const [inline]

7.53.2.39 void optix::VariableObj::getMatrix3x2 (
    bool transpose,
    float * m ) const [inline]

7.53.2.40 void optix::VariableObj::getMatrix3x3 (
    bool transpose,
    float * m ) const [inline]

7.53.2.41 void optix::VariableObj::getMatrix3x4 (
    bool transpose,
    float * m ) const [inline]

7.53.2.42 void optix::VariableObj::getMatrix4x2 (
    bool transpose,
```

```
    float * m ) const [inline]
```

7.53.2.43 void optix::VariableObj::getMatrix4x3 (
 bool transpose,
 float * *m*) const [inline]

7.53.2.44 void optix::VariableObj::getMatrix4x4 (
 bool transpose,
 float * *m*) const [inline]

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

Retrieve the name of the variable.

7.53.2.46 optix::Program optix::VariableObj::getProgram () const [inline]

7.53.2.47 optix::Selector optix::VariableObj::getSelector () const [inline]

7.53.2.48 RTsize optix::VariableObj::getSize () const [inline]

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

7.53.2.49 optix::TextureSampler optix::VariableObj::getTextureSampler () const [inline]

7.53.2.50 optix::Transform optix::VariableObj::getTransform () const [inline]

7.53.2.51 RTobjecttype optix::VariableObj::getType () const [inline]

Query the object type of the variable.

7.53.2.52 unsigned optix::VariableObj::getUint () const [inline]

7.53.2.53 void optix::VariableObj::getUint (
 unsigned & *u1*) const [inline]

7.53.2.54 void optix::VariableObj::getUint (
 unsigned & *u1*,
 unsigned & *u2*) const [inline]

7.53.2.55 void optix::VariableObj::getUint (
 unsigned & *u1*,
 unsigned & *u2*,
 unsigned & *u3*) const [inline]

7.53.2.56 void optix::VariableObj::getUint (
 unsigned & *u1*,
 unsigned & *u2*,
 unsigned & *u3*,

```
    unsigned & u4 ) const [inline]

7.53.2.57 optix::uint2 optix::VariableObj::getUInt2( ) const [inline]

7.53.2.58 optix::uint3 optix::VariableObj::getUInt3( ) const [inline]

7.53.2.59 optix::uint4 optix::VariableObj::getUInt4( ) const [inline]

7.53.2.60 unsigned long long optix::VariableObj::getULongLong( ) const [inline]

7.53.2.61 void optix::VariableObj::getULongLong(
    unsigned long long & ull1 ) const [inline]

7.53.2.62 void optix::VariableObj::getULongLong (
    unsigned long long & ull1,
    unsigned long long & ull2 ) const [inline]

7.53.2.63 void optix::VariableObj::getULongLong (
    unsigned long long & ull1,
    unsigned long long & ull2,
    unsigned long long & ull3 ) const [inline]

7.53.2.64 void optix::VariableObj::getULongLong (
    unsigned long long & ull1,
    unsigned long long & ull2,
    unsigned long long & ull3,
    unsigned long long & ull4 ) const [inline]

7.53.2.65 optix::ulonglong2 optix::VariableObj::getULongLong2( ) const [inline]

7.53.2.66 optix::ulonglong3 optix::VariableObj::getULongLong3( ) const [inline]

7.53.2.67 optix::ulonglong4 optix::VariableObj::getULongLong4( ) const [inline]

7.53.2.68 void optix::VariableObj::getUserData (
    RTsize size,
    void * ptr ) const [inline]
```

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

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

For backwards compatibility. Use [Exception::makeException](#) instead.

7.53.2.70 int optix::APIObj::removeReference() [inline], [inherited]

Decrement the reference count for this object.

7.53.2.71 void optix::VariableObj::set(
Buffer *buffer*) [inline]**7.53.2.72 void optix::VariableObj::set(**
TextureSampler *texturesample*) [inline]**7.53.2.73 void optix::VariableObj::set(**
GeometryGroup *group*) [inline]**7.53.2.74 void optix::VariableObj::set(**
Group *group*) [inline]**7.53.2.75 void optix::VariableObj::set(**
Program *program*) [inline]**7.53.2.76 void optix::VariableObj::set(**
Selector *selector*) [inline]**7.53.2.77 void optix::VariableObj::set(**
Transform *transform*) [inline]**7.53.2.78 void optix::VariableObj::set1fv(**
const float * *f*) [inline]

Set variable value to a scalar float.

7.53.2.79 void optix::VariableObj::set1iv(
const int * *i*) [inline]**7.53.2.80 void optix::VariableObj::set1llv(**
const long long * *i*) [inline]**7.53.2.81 void optix::VariableObj::set1uiv(**
const unsigned int * *u*) [inline]**7.53.2.82 void optix::VariableObj::set1ullv(**
const unsigned long long * *u*) [inline]**7.53.2.83 void optix::VariableObj::set2fv(**
const float * *f*) [inline]

Set variable value to a float2.

7.53.2.84 void optix::VariableObj::set2iv(

const int * *i*) [inline]

7.53.2.85 void optix::VariableObj::set2llv (
const long long * *i*) [inline]

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

7.53.2.87 void optix::VariableObj::set2ullv (
const unsigned long long * *u*) [inline]

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

Set variable value to a float3.

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

7.53.2.90 void optix::VariableObj::set3llv (
const long long * *i*) [inline]

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

7.53.2.92 void optix::VariableObj::set3ullv (
const unsigned long long * *u*) [inline]

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

Set variable value to a float4.

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

7.53.2.95 void optix::VariableObj::set4llv (
const long long * *i*) [inline]

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

7.53.2.97 void optix::VariableObj::set4ullv (
const unsigned long long * *u*) [inline]

7.53.2.98 void optix::VariableObj::setBuffer (

Buffer *buffer*) [inline]

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

Set variable value to a scalar float.

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

Set variable value to a float2.

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

Set variable value to a float2.

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

Set variable value to a float3.

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

Set variable value to a float3.

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

Set variable value to a float4.

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

Set variable value to a float4.

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

7.53.2.107 void optix::VariableObj::setInt (
int *i1*,

```
    int i2 ) [inline]

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

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

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

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

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

7.53.2.113 void optix::VariableObj::setLongLong (
    long long i1 ) [inline]

7.53.2.114 void optix::VariableObj::setLongLong (
    long long i1,
    long long i2 ) [inline]

7.53.2.115 void optix::VariableObj::setLongLong (
    long long i1,
    long long i2,
    long long i3 ) [inline]

7.53.2.116 void optix::VariableObj::setLongLong (
    long long i1,
    long long i2,
    long long i3,
    long long i4 ) [inline]

7.53.2.117 void optix::VariableObj::setLongLong (
    optix::longlong2 i ) [inline]

7.53.2.118 void optix::VariableObj::setLongLong (
```

- ```
 optix::longlong3 i) [inline]

7.53.2.119 void optix::VariableObj::setLongLong (
 optix::longlong4 i) [inline]

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

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

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

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

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

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

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

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

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

7.53.2.129 void optix::VariableObj::setProgramId (
 Program program) [inline]

7.53.2.130 void optix::VariableObj::setTextureSampler (
```

**TextureSampler *texturesample* ) [inline]**

**7.53.2.131 void optix::VariableObj::setUint (**  
**unsigned int *u1* ) [inline]**

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

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

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

**7.53.2.135 void optix::VariableObj::setUint (**  
**optix::uint2 *u* ) [inline]**

**7.53.2.136 void optix::VariableObj::setUint (**  
**optix::uint3 *u* ) [inline]**

**7.53.2.137 void optix::VariableObj::setUint (**  
**optix::uint4 *u* ) [inline]**

**7.53.2.138 void optix::VariableObj::setULongLong (**  
**unsigned long long *u1* ) [inline]**

**7.53.2.139 void optix::VariableObj::setULongLong (**  
**unsigned long long *u1*,**  
**unsigned long long *u2* ) [inline]**

**7.53.2.140 void optix::VariableObj::setULongLong (**  
**unsigned long long *u1*,**  
**unsigned long long *u2*,**  
**unsigned long long *u3* ) [inline]**

**7.53.2.141 void optix::VariableObj::setULongLong (**  
**unsigned long long *u1*,**  
**unsigned long long *u2*,**  
**unsigned long long *u3*,**

**unsigned long long u4 ) [inline]**

**7.53.2.142 void optix::VariableObj::setULongLong (**  
    **optix::ulonglong2 u ) [inline]**

**7.53.2.143 void optix::VariableObj::setULongLong (**  
    **optix::ulonglong3 u ) [inline]**

**7.53.2.144 void optix::VariableObj::setULongLong (**  
    **optix::ulonglong4 u ) [inline]**

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

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

## 7.54 optix::VectorDim< DIM > Struct Template Reference

### 7.55 optix::VectorDim< 2 > Struct Template Reference

#### Public Types

- **typedef float2 VectorType**

#### 7.55.1 Member Typedef Documentation

##### 7.55.1.1 **typedef float2 optix::VectorDim< 2 >::VectorType**

### 7.56 optix::VectorDim< 3 > Struct Template Reference

#### Public Types

- **typedef float3 VectorType**

#### 7.56.1 Member Typedef Documentation

##### 7.56.1.1 **typedef float3 optix::VectorDim< 3 >::VectorType**

### 7.57 optix::VectorDim< 4 > Struct Template Reference

#### Public Types

- **typedef float4 VectorType**

### 7.57.1 Member Typedef Documentation

#### 7.57.1.1 `typedef float4 optix::VectorDim< 4 >::VectorType`

### 7.58 optix::VectorTypes< T, Dim > Struct Template Reference

### 7.59 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)`

#### 7.59.1 Member Typedef Documentation

##### 7.59.1.1 `typedef float optix::VectorTypes< float, 1 >::Type`

#### 7.59.2 Member Function Documentation

##### 7.59.2.1 `template<class S > static __device__ __forceinline__ Type optix::VectorTypes< float,` `1 >::make (` `S s ) [inline], [static]`

### 7.60 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)`

### 7.60.1 Member Typedef Documentation

#### 7.60.1.1 `typedef float2 optix::VectorTypes< float, 2 >::Type`

### 7.60.2 Member Function Documentation

```
7.60.2.1 template<class S> static __device__ __forceinline__ Type optix::VectorTypes< float,
2 >::make(
 S s) [inline], [static]
```

## 7.61 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)`

### 7.61.1 Member Typedef Documentation

#### 7.61.1.1 `typedef float3 optix::VectorTypes< float, 3 >::Type`

### 7.61.2 Member Function Documentation

```
7.61.2.1 template<class S> static __device__ __forceinline__ Type optix::VectorTypes< float,
3 >::make(
 S s) [inline], [static]
```

## 7.62 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)`

### 7.62.1 Member Typedef Documentation

#### 7.62.1.1 `typedef float4 optix::VectorTypes< float, 4 >::Type`

### 7.62.2 Member Function Documentation

```
7.62.2.1 template<class S> static __device__ __forceinline__ Type optix::VectorTypes< float,
4 >::make (
 S s) [inline], [static]
```

## 7.63 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)`

### 7.63.1 Member Typedef Documentation

#### 7.63.1.1 `typedef int optix::VectorTypes< int, 1 >::Type`

### 7.63.2 Member Function Documentation

```
7.63.2.1 template<class S> static __device__ __forceinline__ Type optix::VectorTypes< int, 1
>::make (
 S s) [inline], [static]
```

## 7.64 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)`

### 7.64.1 Member Typedef Documentation

#### 7.64.1.1 `typedef int2 optix::VectorTypes< int, 2 >::Type`

### 7.64.2 Member Function Documentation

```
7.64.2.1 template<class S> static __device__ __forceinline__ Type optix::VectorTypes< int, 2
>::make (
 S s) [inline], [static]
```

## 7.65 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)
```

### 7.65.1 Member Typedef Documentation

#### 7.65.1.1 `typedef int3 optix::VectorTypes< int, 3 >::Type`

### 7.65.2 Member Function Documentation

```
7.65.2.1 template<class S> static __device__ __forceinline__ Type optix::VectorTypes< int, 3
>::make (
 S s) [inline], [static]
```

## 7.66 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)
```

### 7.66.1 Member Typedef Documentation

#### 7.66.1.1 `typedef int4 optix::VectorTypes< int, 4 >::Type`

### 7.66.2 Member Function Documentation

```
7.66.2.1 template<class S> static __device__ __forceinline__ Type optix::VectorTypes< int, 4
>::make (
 S s) [inline], [static]
```

## 7.67 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)`

### 7.67.1 Member Typedef Documentation

#### 7.67.1.1 `typedef unsigned int optix::VectorTypes< unsigned int, 1 >::Type`

### 7.67.2 Member Function Documentation

```
7.67.2.1 static __device__ __forceinline__ Type optix::VectorTypes< unsigned int, 1 >::make (
 unsigned int s) [inline], [static]
```

```
7.67.2.2 template<class S> static __device__ __forceinline__ Type optix::VectorTypes<
 unsigned int, 1 >::make (
 S s) [inline], [static]
```

## 7.68 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)
```

### 7.68.1 Member Typedef Documentation

#### 7.68.1.1 typedef uint2 optix::VectorTypes< unsigned int, 2 >::Type

### 7.68.2 Member Function Documentation

```
7.68.2.1 template<class S > static __device__ __forceinline__ Type optix::VectorTypes<
unsigned int, 2 >::make (
 S s) [inline], [static]
```

## 7.69 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 \(S s\)](#)

### 7.69.1 Member Typedef Documentation

#### 7.69.1.1 typedef uint3 optix::VectorTypes< unsigned int, 3 >::Type

### 7.69.2 Member Function Documentation

```
7.69.2.1 template<class S > static __device__ __forceinline__ Type optix::VectorTypes<
unsigned int, 3 >::make (
 S s) [inline], [static]
```

## 7.70 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 \(S s\)](#)

### 7.70.1 Member Typedef Documentation

#### 7.70.1.1 `typedef uint4 optix::VectorTypes< unsigned int, 4 >::Type`

### 7.70.2 Member Function Documentation

#### 7.70.2.1 `template<class S> static __device__ __forceinline__ Type optix::VectorTypes< unsigned int, 4 >::make ( S s ) [inline], [static]`

## 8 File Documentation

### 8.1 Atom.h File Reference

#### Namespaces

- [optix](#)
- [optix::prime](#)

#### Constant Groups

- [optix](#)
- [optix::prime](#)

#### Macros

- [#define PRIME\\_ATOM32\\_GCC](#)

#### Typedefs

- [typedef unsigned int atomic\\_word](#)

### 8.1.1 Macro Definition Documentation

#### 8.1.1.1 #define PRIME\_ATOM32\_GCC

### 8.1.2 Typedef Documentation

#### 8.1.2.1 typedef unsigned int atomic\_word

## 8.2 doxygen\_hierarchy.h File Reference

### 8.3 footer.tex File Reference

### 8.4 Handle.h File Reference

#### Namespaces

- optix
- optix::prime

#### Constant Groups

- optix
- optix::prime

## 8.5 header.tex File Reference

#### Variables

- Latex header for doxygen documentclass
- Latex header for doxygen showboxes option to debug usepackage [T1]

### 8.5.1 Variable Documentation

#### 8.5.1.1 Latex header for doxygen documentclass

#### 8.5.1.2 Latex header for doxygen showboxes option to debug usepackage[T1]

**Initial value:**

```
{fontenc}
% \pdfmapfile{+winfonts.map}
% \renewcommand\rmdefault{trebuchet} CUDA documentation uses this
```

## 8.6 interop\_types.h File Reference

### 8.7 nvoptix.h File Reference

#### Classes

- struct OptiXAPI

#### Macros

- #define NVOPTIX\_ABI\_VERSION 12

#### Typedefs

- typedef RTResult(\* PRTACCELERATIONCREATE )(RTcontext context\_api, RTacceleration \*acceleration)
- typedef RTResult(\* PRTACCELERATIONDESTROY )(RTacceleration acceleration\_api)
- typedef RTResult(\* PRTACCELERATIONGETBUILDER )(RTacceleration acceleration\_api, const char \*\*return\_string)
- typedef RTResult(\* PRTACCELERATIONGETCONTEXT )(RTacceleration acceleration\_api, RTcontext \*c)
- typedef RTResult(\* PRTACCELERATIONGETDATA )(RTacceleration acceleration\_api, void \*data)
- typedef RTResult(\* PRTACCELERATIONGETDATASIZE )(RTacceleration acceleration\_api, RTsize \*size)
- typedef RTResult(\* PRTACCELERATIONGETPROPERTY )(RTacceleration acceleration\_api, const char \*name, const char \*\*return\_string)
- typedef RTResult(\* PRTACCELERATIONGETTRaverser )(RTacceleration acceleration\_api, const char \*\*return\_string)
- typedef RTResult(\* PRTACCELERATIONISDIRTY )(RTacceleration acceleration\_api, int \*dirty)
- typedef RTResult(\* PRTACCELERATIONMARKDIRTY )(RTacceleration acceleration\_api)
- typedef RTResult(\* PRTACCELERATIONSETBUILDER )(RTacceleration acceleration\_api, const char \*builder)
- typedef RTResult(\* PRTACCELERATIONSETDATA )(RTacceleration acceleration\_api, const void \*data, RTsize size)
- typedef RTResult(\* PRTACCELERATIONsetProperty )(RTacceleration acceleration\_api, const char \*name, const char \*value)
- typedef RTResult(\* PRTACCELERATIONSETTRaverser )(RTacceleration acceleration\_api, const char \*traverser)
- typedef RTResult(\* PRTACCELERATIONVALIDATE )(RTacceleration acceleration\_api)
- typedef RTResult(\* PRTBUFFERCREATE )(RTcontext context\_api, unsigned int type, RTbuffer \*buffer)
- typedef RTResult(\* PRTBUFFERCREATEFORCUDA )(RTcontext context\_api, unsigned int type, RTbuffer \*buffer)
- typedef RTResult(\* PRTBUFFERCREATEFROMGLBO )(RTcontext context\_api, unsigned int type, unsigned int gl\_id, RTbuffer \*buffer)
- typedef RTResult(\* PRTBUFFERDESTROY )(RTbuffer buffer\_api)

- `typedef RTResult(* PRTBUFFERGLREGISTER )(RTbuffer buffer_api)`
- `typedef RTResult(* PRTBUFFERGLUNREGISTER )(RTbuffer buffer_api)`
- `typedef RTResult(* PRTBUFFERGETCONTEXT )(RTbuffer buffer_api, RTcontext *c)`
- `typedef RTResult(* PRTBUFFERGETDEVICEPOINTER )(RTbuffer buffer_api, int optix_device_ordinal, void **device_pointer)`
- `typedef RTResult(* PRTBUFFERGETDIMENSIONALITY )(RTbuffer buffer_api, unsigned int *dimensionality)`
- `typedef RTResult(* PRTBUFFERGETELEMENTSIZE )(RTbuffer buffer_api, RTsize *size_of_element)`
- `typedef RTResult(* PRTBUFFERGETFORMAT )(RTbuffer buffer_api, RTformat *format)`
- `typedef RTResult(* PRTBUFFERGETGLBOID )(RTbuffer buffer_api, unsigned int *gl_id)`
- `typedef RTResult(* PRTBUFFERGETID )(RTbuffer buffer_api, int *buffer_id)`
- `typedef RTResult(* PRTBUFFERGETMIPLEVELCOUNT )(RTbuffer buffer_api, unsigned int *level)`
- `typedef RTResult(* PRTBUFFERGETMIPLEVELSIZE1D )(RTbuffer buffer_api, unsigned int level, RTsize *width)`
- `typedef RTResult(* PRTBUFFERGETMIPLEVELSIZE2D )(RTbuffer buffer_api, unsigned int level, RTsize *width, RTsize *height)`
- `typedef RTResult(* PRTBUFFERGETMIPLEVELSIZE3D )(RTbuffer buffer_api, unsigned int level, RTsize *width, RTsize *height, RTsize *depth)`
- `typedef RTResult(* PRTBUFFERGETSIZE1D )(RTbuffer buffer_api, RTsize *width)`
- `typedef RTResult(* PRTBUFFERGETSIZE2D )(RTbuffer buffer_api, RTsize *width, RTsize *height)`
- `typedef RTResult(* PRTBUFFERGETSIZE3D )(RTbuffer buffer_api, RTsize *width, RTsize *height, RTsize *depth)`
- `typedef RTResult(* PRTBUFFERGETSIZEV )(RTbuffer buffer_api, unsigned int maxdim, RTsize *outdims)`
- `typedef RTResult(* PRTBUFFERMAP )(RTbuffer buffer_api, void **user_pointer)`
- `typedef RTResult(* PRTBUFFERMAPEX )(RTbuffer buffer, unsigned int map_flags, unsigned int level, void *user_owned, void **optix_owned)`
- `typedef RTResult(* PRTBUFFERMARKDIRTY )(RTbuffer buffer_api)`
- `typedef RTResult(* PRTBUFFERSETDEVICEPOINTER )(RTbuffer buffer_api, int optix_device_ordinal, void *device_pointer)`
- `typedef RTResult(* PRTBUFFERSETELEMENTSIZE )(RTbuffer buffer_api, RTsize size_of_element)`
- `typedef RTResult(* PRTBUFFERSETFORMAT )(RTbuffer buffer_api, RTformat type)`
- `typedef RTResult(* PRTBUFFERSETMIPLEVELCOUNT )(RTbuffer buffer_api, unsigned int levels)`
- `typedef RTResult(* PRTBUFFERSETSIZE1D )(RTbuffer buffer_api, RTsize width)`
- `typedef RTResult(* PRTBUFFERSETSIZE2D )(RTbuffer buffer_api, RTsize width, RTsize height)`
- `typedef RTResult(* PRTBUFFERSETSIZE3D )(RTbuffer buffer_api, RTsize width, RTsize height, RTsize depth)`
- `typedef RTResult(* PRTBUFFERSETSIZEV )(RTbuffer buffer_api, unsigned int dimensionality, const RTsize *indims)`
- `typedef RTResult(* PRTBUFFERUNMAP )(RTbuffer buffer_api)`
- `typedef RTResult(* PRTBUFFERUNMAPEX )(RTbuffer buffer_api, unsigned int level)`
- `typedef RTResult(* PRTBUFFERVALIDATE )(RTbuffer buffer_api)`

- `typedef RTResult(* PRTBUFFERGETPROGRESSIVEUPDATEREADY )(RTbuffer buffer_api, int *ready, unsigned int *subframe_count, unsigned int *max_subframes)`
- `typedef RTResult(* PRTBUFFERBINDPROGRESSIVESTREAM )(RTbuffer buffer_api, RTbuffer source)`
- `typedef RTResult(* PRTBUFFERSETATTRIBUTE )(RTbuffer buffer, RTbufferattribute attrib, RTsize size, const void *p)`
- `typedef RTResult(* PRTBUFFERGETATTRIBUTE )(RTbuffer buffer, RTbufferattribute attrib, RTsize size, void *p)`
- `typedef RTResult(* PRTCMDLISTCREATE )(RTcontext context_api, RTcommandlist *list)`
- `typedef RTResult(* PRTCMDLISTDESTROY )(RTcommandlist list_api)`
- `typedef RTResult(* PRTCMDLISTAPPENDPOSTPROCESSINGSTAGE )(RTcommandlist list_api, RTpostprocessingstage stage_api, RTsize launch_width, RTsize launch_height)`
- `typedef RTResult(* PRTCMDLISTAPPENDLAUNCH2D )(RTcommandlist list_api, unsigned int entry_point_index, RTsize launch_width, RTsize launch_height)`
- `typedef RTResult(* PRTCMDLISTFINALIZE )(RTcommandlist list_api)`
- `typedef RTResult(* PRTCMDLISTEXECUTE )(RTcommandlist list_api)`
- `typedef RTResult(* PRTCMDLISTGETCONTEXT )(RTcommandlist list_api, RTcontext *c)`
- `typedef RTResult(* PRTCONTEXTCOMPILE )(RTcontext context_api)`
- `typedef RTResult(* PRTCONTEXTCREATE )(RTcontext *context)`
- `typedef RTResult(* PRTCONTEXTDECLAREVARIABLE )(RTcontext context_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTCONTEXTDESTROY )(RTcontext context_api)`
- `typedef RTResult(* PRTCONTEXTSETATTRIBUTE )(RTcontext context_api, RTcontextattribute attrib, RTsize size, const void *p)`
- `typedef RTResult(* PRTCONTEXTGETATTRIBUTE )(RTcontext context_api, RTcontextattribute attrib, RTsize size, void *p)`
- `typedef RTResult(* PRTCONTEXTGETBUFFERFROMID )(RTcontext context_api, int buffer_id, RTbuffer *buffer)`
- `typedef RTResult(* PRTCONTEXTGETDEVICECOUNT )(RTcontext context_api, unsigned int *count)`
- `typedef RTResult(* PRTCONTEXTGETDEVICES )(RTcontext context_api, int *devices)`
- `typedef RTResult(* PRTCONTEXTGETENTRYPOINTCOUNT )(RTcontext context_api, unsigned int *num_entry_points)`
- `typedef void(* PRTCONTEXTGETERRORSTRING )(RTcontext context_api, RTResult code, const char **return_string)`
- `typedef RTResult(* PRTCONTEXTGETEXCEPTIONENABLED )(RTcontext context_api, RTexception exception, int *enabled)`
- `typedef RTResult(* PRTCONTEXTGETEXCEPTIONPROGRAM )(RTcontext context_api, unsigned int entry_point_index, RTprogram *program)`
- `typedef RTResult(* PRTCONTEXTGETMAXCALLABLEPROGRAMDEPTH )(RTcontext context_api, unsigned int *max_depth)`
- `typedef RTResult(* PRTCONTEXTGETMAXTRACEDEPTH )(RTcontext context_api, unsigned int *max_depth)`
- `typedef RTResult(* PRTCONTEXTGETMISSPROGRAM )(RTcontext context_api, unsigned int ray_type_index, RTprogram *program)`
- `typedef RTResult(* PRTCONTEXTGETPRINTBUFFERSIZE )(RTcontext context_api, RTsize *buffer_size_bytes)`

- `typedef RTResult(* PRTCONTEXTGETPRINTENABLED )(RTcontext context_api, int *enabled)`
- `typedef RTResult(* PRTCONTEXTGETPRINTLAUNCHINDEX )(RTcontext context_api, int *x, int *y, int *z)`
- `typedef RTResult(* PRTCONTEXTGETPROGRAMFROMID )(RTcontext context_api, int program_id, RTprogram *program)`
- `typedef RTResult(* PRTCONTEXTGETRAYGENERATIONPROGRAM )(RTcontext context_api, unsigned int entry_point_index, RTprogram *program)`
- `typedef RTResult(* PRTCONTEXTGETRAYTYPECOUNT )(RTcontext context_api, unsigned int *num_ray_types)`
- `typedef RTResult(* PRTCONTEXTGETRUNNINGSTATE )(RTcontext context_api, int *running)`
- `typedef RTResult(* PRTCONTEXTGETSTACKSIZE )(RTcontext context_api, RTsize *stack_size_bytes)`
- `typedef RTResult(* PRTCONTEXTGETTEXTURESAMPLERFROMID )(RTcontext context_api, int sampler_id, RTtexturesampler *sampler)`
- `typedef RTResult(* PRTCONTEXTGETVARIABLE )(RTcontext context_api, unsigned int index, RTvariable *v)`
- `typedef RTResult(* PRTCONTEXTGETVARIABLECOUNT )(RTcontext context_api, unsigned int *c)`
- `typedef RTResult(* PRTCONTEXTLAUNCH1D )(RTcontext context_api, unsigned int entry_point_index, RTsize image_width)`
- `typedef RTResult(* PRTCONTEXTLAUNCH2D )(RTcontext context_api, unsigned int entry_point_index, RTsize image_width, RTsize image_height)`
- `typedef RTResult(* PRTCONTEXTLAUNCH3D )(RTcontext context_api, unsigned int entry_point_index, RTsize image_width, RTsize image_height, RTsize image_depth)`
- `typedef RTResult(* PRTCONTEXTLAUNCHPROGRESSIVE2D )(RTcontext context_api, unsigned int entry_point_index, RTsize image_width, RTsize image_height, unsigned int max_subframes)`
- `typedef RTResult(* PRTCONTEXTSTOPPROGRESSIVE )(RTcontext context_api)`
- `typedef RTResult(* PRTCONTEXTQUERYVARIABLE )(RTcontext context_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTCONTEXTREMOVEVARIABLE )(RTcontext context_api, RTvariable v_api)`
- `typedef RTResult(* PRTCONTEXTSETDEVICES )(RTcontext context_api, unsigned int count, const int *devices)`
- `typedef RTResult(* PRTCONTEXTSETENTRYPOINTCOUNT )(RTcontext context_api, unsigned int num_entry_points)`
- `typedef RTResult(* PRTCONTEXTSETEXCEPTIONENABLED )(RTcontext context_api, RTexception exception, int enabled)`
- `typedef RTResult(* PRTCONTEXTSETEXCEPTIONPROGRAM )(RTcontext context_api, unsigned int entry_point_index, RTprogram program_api)`
- `typedef RTResult(* PRTCONTEXTSETMAXCALLABLEPROGRAMDEPTH )(RTcontext context_api, unsigned int max_depth)`
- `typedef RTResult(* PRTCONTEXTSETMAXTRACEDEPTH )(RTcontext context_api, unsigned int max_depth)`
- `typedef RTResult(* PRTCONTEXTSETMISSPROGRAM )(RTcontext context_api, unsigned int ray_type_index, RTprogram program_api)`
- `typedef RTResult(* PRTCONTEXTSETPRINTBUFFERSIZE )(RTcontext context_api, RTsize buffer_size_bytes)`

- `typedef RTResult(* PRTCONTEXTSETPRINTENABLED )(RTcontext context_api, int enabled)`
- `typedef RTResult(* PRTCONTEXTSETPRINTLAUNCHINDEX )(RTcontext context_api, int x, int y, int z)`
- `typedef RTResult(* PRTCONTEXTSETRAYGENERATIONPROGRAM )(RTcontext context_api, unsigned int entry_point_index, RTprogram program_api)`
- `typedef RTResult(* PRTCONTEXTSETRAYTYPECOUNT )(RTcontext context_api, unsigned int num_ray_types)`
- `typedef RTResult(* PRTCONTEXTSETSTACKSIZE )(RTcontext context_api, RTsize stack_size_bytes)`
- `typedef RTResult(* PRTCONTEXTSETTIMEOUTCALLBACK )(RTcontext context_api, RTtimeoutcallback callback, double seconds)`
- `typedef RTResult(* PRTCONTEXTSETUSAGEREPORTCALLBACK )(RTcontext context_api, RTusagereportcallback callback, int verbosity, void *cbdata)`
- `typedef RTResult(* PRTCONTEXTVALIDATE )(RTcontext context_api)`
- `typedef RTResult(* PRTDEVICEGETATTRIBUTE )(int ordinal, RTdeviceattribute attrib, RTsize size, void *p)`
- `typedef RTResult(* PRTDEVICEGETDEVICECOUNT )(unsigned int *count)`
- `typedef RTResult(* PRTGEOMETRYCREATE )(RTcontext context_api, RTgeometry *geometry)`
- `typedef RTResult(* PRTGEOMETRYDECLAREVARIABLE )(RTgeometry geometry_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTGEOMETRYDESTROY )(RTgeometry geometry_api)`
- `typedef RTResult(* PRTGEOMETRYGETBOUNDINGBOXPROGRAM )(RTgeometry geometry_api, RTprogram *program)`
- `typedef RTResult(* PRTGEOMETRYGETCONTEXT )(RTgeometry geometry_api, RTcontext *c)`
- `typedef RTResult(* PRTGEOMETRYGETINTERSECTIONPROGRAM )(RTgeometry geometry_api, RTprogram *program)`
- `typedef RTResult(* PRTGEOMETRYGETPRIMITIVECOUNT )(RTgeometry geometry_api, unsigned int *num_primitives)`
- `typedef RTResult(* PRTGEOMETRYGETPRIMITIVEINDEXOFFSET )(RTgeometry geometry_api, unsigned int *index_offset)`
- `typedef RTResult(* PRTGEOMETRYGETVARIABLE )(RTgeometry geometry_api, unsigned int index, RTvariable *v)`
- `typedef RTResult(* PRTGEOMETRYGETVARIABLECOUNT )(RTgeometry geometry_api, unsigned int *c)`
- `typedef RTResult(* PRTGEOMETRYGROUPCREATE )(RTcontext context_api, RTgeometrygroup *geometrygroup)`
- `typedef RTResult(* PRTGEOMETRYGROUPDESTROY )(RTgeometrygroup geometrygroup_api)`
- `typedef RTResult(* PRTGEOMETRYGROUPGETACCELERATION )(RTgeometrygroup geometrygroup_api, RTacceleration *acceleration)`
- `typedef RTResult(* PRTGEOMETRYGROUPGETCHILD )(RTgeometrygroup geometrygroup_api, unsigned int index, RTgeometryinstance *geometryinstance)`
- `typedef RTResult(* PRTGEOMETRYGROUPGETCHILDCOUNT )(RTgeometrygroup geometrygroup_api, unsigned int *count)`
- `typedef RTResult(* PRTGEOMETRYGROUPGETCONTEXT )(RTgeometrygroup geometrygroup_api, RTcontext *c)`
- `typedef RTResult(* PRTGEOMETRYGROUPSETACCELERATION )(RTgeometrygroup geometrygroup_api, RTacceleration acceleration_api)`

- `typedef RTResult(* PRTGEOMETRYGROUPSETCHILD )(RTgeometrygroup geometrygroup_api, unsigned int index, RTgeometryinstance geometryinstance_api)`
- `typedef RTResult(* PRTGEOMETRYGROUPSETCHILDCOUNT )(RTgeometrygroup geometrygroup_api, unsigned int count)`
- `typedef RTResult(* PRTGEOMETRYGROUPVALIDATE )(RTgeometrygroup geometrygroup_api)`
- `typedef RTResult(* PRTGEOMETRYINSTANCECREATE )(RTcontext context_api, RTgeometryinstance *geometryinstance)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEDECLAREVARIABLE )(RTgeometryinstance gi_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEDESTROY )(RTgeometryinstance geometryinstance_api)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEGETCONTEXT )(RTgeometryinstance gi_api, RTcontext *c)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEGETGEOMETRY )(RTgeometryinstance gi_api, RTgeometry *geo)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEGETGEOMETRYTRIANGLES )(RTgeometryinstance gi_api, RTgeometrytriangles *geo)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEGETMATERIAL )(RTgeometryinstance gi_api, unsigned int material_idx, RTmaterial *mat)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEGETMATERIALCOUNT )(RTgeometryinstance gi_api, unsigned int *num_materials)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEGETVARIABLE )(RTgeometryinstance gi_api, unsigned int index, RTvariable *v)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEGETVARIABLECOUNT )(RTgeometryinstance gi_api, unsigned int *c)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEQUERYVARIABLE )(RTgeometryinstance gi_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEREMOVEVARIABLE )(RTgeometryinstance gi_api, RTvariable v_api)`
- `typedef RTResult(* PRTGEOMETRYINSTANCESETGEOMETRY )(RTgeometryinstance gi_api, RTgeometry geo_api)`
- `typedef RTResult(* PRTGEOMETRYINSTANCESETGEOMETRYTRIANGLES )(RTgeometryinstance gi_api, RTgeometrytriangles geo_api)`
- `typedef RTResult(* PRTGEOMETRYINSTANCESETMATERIAL )(RTgeometryinstance gi_api, unsigned int material_idx, RTmaterial mat_api)`
- `typedef RTResult(* PRTGEOMETRYINSTANCESETMATERIALCOUNT )(RTgeometryinstance gi_api, unsigned int num_materials)`
- `typedef RTResult(* PRTGEOMETRYINSTANCEVALIDATE )(RTgeometryinstance geometryinstance_api)`
- `typedef RTResult(* PRTGEOMETRYISDIRTY )(RTgeometry geometry_api, int *dirty)`
- `typedef RTResult(* PRTGEOMETRYMARKDIRTY )(RTgeometry geometry_api)`
- `typedef RTResult(* PRTGEOMETRYQUERYVARIABLE )(RTgeometry geometry_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTGEOMETRYREMOVEVARIABLE )(RTgeometry geometry_api, RTvariable v_api)`
- `typedef RTResult(* PRTGEOMETRYSETBOUNDINGBOXPROGRAM )(RTgeometry geometry_api, RTprogram program_api)`

- `typedef RTResult(* PRTGEOMETRYSETINTERSECTIONPROGRAM )(RTgeometry geometry_api, RTprogram program_api)`
- `typedef RTResult(* PRTGEOMETRYSETPRIMITIVECOUNT )(RTgeometry geometry_api, unsigned int num_primitives)`
- `typedef RTResult(* PRTGEOMETRYSETPRIMITIVEINDEXOFFSET )(RTgeometry geometry_api, unsigned int index_offset)`
- `typedef RTResult(* PRTGEOMETRYSETMOTIONRANGE )(RTgeometry geometry_api, float timeBegin, float timeEnd)`
- `typedef RTResult(* PRTGEOMETRYGETMOTIONRANGE )(RTgeometry geometry_api, float *timeBegin, float *timeEnd)`
- `typedef RTResult(* PRTGEOMETRYSETMOTIONBORDERMODE )(RTgeometry geometry_api, RTmotionbordermode beginMode, RTmotionbordermode endMode)`
- `typedef RTResult(* PRTGEOMETRYGETMOTIONBORDERMODE )(RTgeometry geometry_api, RTmotionbordermode *beginMode, RTmotionbordermode *endMode)`
- `typedef RTResult(* PRTGEOMETRYSETMOTIONSTEPS )(RTgeometry geometry_api, unsigned int n)`
- `typedef RTResult(* PRTGEOMETRYGETMOTIONSTEPS )(RTgeometry geometry_api, unsigned int *n)`
- `typedef RTResult(* PRTGEOMETRYVALIDATE )(RTgeometry geometry_api)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESCREATE )(RTcontext context_api, RTgeometrytriangles *geometrytriangles)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESDESTROY )(RTgeometrytriangles geometrytriangles_api)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESGETCONTEXT )(RTgeometrytriangles geometrytriangles_api, RTcontext *context)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESGETPRIMITIVEINDEXOFFSET )(RTgeometrytriangles geometrytriangles_api, unsigned int *index_offset)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESSETPRIMITIVEINDEXOFFSET )(RTgeometrytriangles geometrytriangles_api, unsigned int index_offset)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESSETPRETRANSFORMMATRIX )(RTgeometrytriangles geometrytriangles_api, int transpose, const float *matrix)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESGETPRETRANSFORMMATRIX )(RTgeometrytriangles geometrytriangles_api, int transpose, float *matrix)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESSETPRIMITIVECOUNT )(RTgeometrytriangles geometrytriangles_api, unsigned int num_triangles)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESGETPRIMITIVECOUNT )(RTgeometrytriangles geometrytriangles_api, unsigned int *num_triangles)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESSETINDEXEDTRIANGLESDEPRECATED )(RTgeometrytriangles geometrytriangles_api, unsigned int num_triangles, RTbuffer index_buffer_api, RTsize index_buffer_byte_offset, RTsize tri_indices_byte_stride, RTformat tri_indices_format, unsigned int num_vertices, RTbuffer vertex_buffer_api, RTsize vertex_buffer_byte_offset, RTsize vertex_byte_stride, RTformat position_format, RTgeometrybuildflags build_flags)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESSETTRIANGLESDEPRECATED )(RTgeometrytriangles geometrytriangles_api, unsigned int num_triangles, RTbuffer vertex_buffer_api, RTsize vertex_buffer_byte_offset, RTsize vertex_byte_stride, RTformat position_format, RTgeometrybuildflags build_flags)`

- **typedef RTResult(\* PRTGEOMETRYTRIANGLESSETTRIANGLESINDICES**  
)(**RTgeometrytriangles** geometrytriangles\_api, **RTbuffer** index\_buffer\_api, **RTsize**  
**index\_buffer\_byte\_offset**, **RTsize** tri\_indices\_byte\_stride, **RTformat** tri\_indices\_format)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESSETVERTICES** )(**RTgeometrytriangles**  
geometrytriangles\_api, unsigned int num\_vertices, **RTbuffer** vertex\_buffer\_api, **RTsize**  
**vertex\_buffer\_byte\_offset**, **RTsize** vertex\_byte\_stride, **RTformat** position\_format)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESSETMOTIONVERTICES** )(**RTgeometrytriangles**  
geometrytriangles\_api, unsigned int num\_vertices, **RTbuffer** vertex\_buffer\_api, **RTsize**  
**vertex\_buffer\_byte\_offset**, **RTsize** vertex\_byte\_stride, **RTsize** vertex\_motion\_step\_byte\_stride,  
**RTformat** position\_format)
- **typedef RTResult(\***  
**PRTGEOMETRYTRIANGLESSETMOTIONVERTEXESMULTIBUFFERDEPRECATED**  
)(**RTgeometrytriangles** geometrytriangles\_api, unsigned int num\_vertices, **RTbuffer**  
**\*vertex\_buffers**, **RTsize** vertex\_buffer\_byte\_offset, **RTsize** vertex\_byte\_stride, **RTformat**  
**position\_format**)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESSETMOTIONVERTEXESMULTIBUFFER**  
)(**RTgeometrytriangles** geometrytriangles\_api, unsigned int num\_vertices, **RTbuffer**  
**\*vertex\_buffers**, unsigned int vertex\_buffer\_count, **RTsize** vertex\_buffer\_byte\_offset, **RTsize**  
**vertex\_byte\_stride**, **RTformat** position\_format)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESSETMOTIONSTEPS** )(**RTgeometrytriangles**  
geometrytriangles\_api, unsigned int num\_motion\_steps)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESGETMOTIONSTEPS** )(**RTgeometrytriangles**  
geometrytriangles\_api, unsigned int \*num\_motion\_steps)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESSETMOTIONRANGE** )(**RTgeometrytriangles**  
geometrytriangles\_api, float timeBegin, float timeEnd)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESGETMOTIONRANGE** )(**RTgeometrytriangles**  
geometrytriangles\_api, float \*timeBegin, float \*timeEnd)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESSETMOTIONBORDERMODE**  
)(**RTgeometrytriangles** geometrytriangles\_api, **RTmotionbordermode** beginMode,  
**RTmotionbordermode** endMode)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESGETMOTIONBORDERMODE**  
)(**RTgeometrytriangles** geometrytriangles\_api, **RTmotionbordermode** \*beginMode,  
**RTmotionbordermode** \*endMode)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESSETBUILDFLAGS** )(**RTgeometrytriangles**  
geometrytriangles, **RTgeometrybuildflags** build\_flags)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESSETMATERIALCOUNT** )(**RTgeometrytriangles**  
geometrytriangles, unsigned int num\_materials)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESGETMATERIALCOUNT** )(**RTgeometrytriangles**  
geometrytriangles, unsigned int \*num\_materials)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESSETMATERIALINDICES** )(**RTgeometrytriangles**  
geometrytriangles, **RTbuffer** material\_index\_buffer, **RTsize** material\_index\_buffer\_byte\_offset,  
**RTsize** material\_index\_byte\_stride, **RTformat** material\_index\_format)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESSETFLAGSPERMATERIAL**  
)(**RTgeometrytriangles** geometrytriangles, unsigned int material\_index, **RTgeometryflags** flags)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESGETFLAGSPERMATERIAL**  
)(**RTgeometrytriangles** geometrytriangles, unsigned int material\_index, **RTgeometryflags** \*flags)
- **typedef RTResult(\* PRTGEOMETRYTRIANGLESVALIDATE** )(**RTgeometrytriangles**  
geometrytriangles\_api)

- `typedef RTResult(* PRTGEOMETRYTRIANGLESSETATTRIBUTEPROGRAM )(RTgeometrytriangles geometrytriangles_api, RTprogram program)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESGETATTRIBUTEPROGRAM )(RTgeometrytriangles geometrytriangles_api, RTprogram *program)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESDECLAREVARIABLE )(RTgeometrytriangles geometrytriangles, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESQUERYVARIABLE )(RTgeometrytriangles geometrytriangles, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESREMOVEVARIABLE )(RTgeometrytriangles geometrytriangles, RTvariable v)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESGETVARIABLECOUNT )(RTgeometrytriangles geometrytriangles, unsigned int *count)`
- `typedef RTResult(* PRTGEOMETRYTRIANGLESGETVARIABLE )(RTgeometrytriangles geometrytriangles, unsigned int index, RTvariable *v)`
- `typedef RTResult(* PRTGETVERSION )(unsigned int *version)`
- `typedef RTResult(* PRTGLOBALSETATTRIBUTE )(RTglobalattribute attrib, RTsize size, const void *p)`
- `typedef RTResult(* PRTGLOBALGETATTRIBUTE )(RTglobalattribute attrib, RTsize size, void *p)`
- `typedef RTResult(* PRTGROUPCREATE )(RTcontext context_api, RTgroup *group)`
- `typedef RTResult(* PRTGROUPDESTROY )(RTgroup group_api)`
- `typedef RTResult(* PRTGROUPGETACCELERATION )(RTgroup group_api, RTacceleration *acceleration)`
- `typedef RTResult(* PRTGROUPGETCHILD )(RTgroup group_api, unsigned int index, RTobject *child)`
- `typedef RTResult(* PRTGROUPGETCHILDCOUNT )(RTgroup group_api, unsigned int *count)`
- `typedef RTResult(* PRTGROUPGETCHILDTYPE )(RTgroup group_api, unsigned int index, RTobjecttype *type)`
- `typedef RTResult(* PRTGROUPGETCONTEXT )(RTgroup group_api, RTcontext *c)`
- `typedef RTResult(* PRTGROUPSETACCELERATION )(RTgroup group_api, RTacceleration acceleration_api)`
- `typedef RTResult(* PRTGROUPSETCHILD )(RTgroup group_api, unsigned int index, RTobject child)`
- `typedef RTResult(* PRTGROUPSETCHILDCOUNT )(RTgroup group_api, unsigned int count)`
- `typedef RTResult(* PRTGROUPVALIDATE )(RTgroup group_api)`
- `typedef RTResult(* PRTGROUPSETVISIBILITYMASK )(RTgroup group, RTvisibilitymask visibility_mask)`
- `typedef RTResult(* PRTGROUPGETVISIBILITYMASK )(RTgroup group, RTvisibilitymask *visibility_mask)`
- `typedef RTResult(* PRTGEOMETRYGROUPSETFLAGS )(RTgeometrygroup geometrygroup, RTinstanceflags instance_flags)`
- `typedef RTResult(* PRTGEOMETRYGROUPGETFLAGS )(RTgeometrygroup geometrygroup, RTinstanceflags *instance_flags)`
- `typedef RTResult(* PRTGEOMETRYGROUPSETVISIBILITYMASK )(RTgeometrygroup geometrygroup, RTvisibilitymask mask)`
- `typedef RTResult(* PRTGEOMETRYGROUPGETVISIBILITYMASK )(RTgeometrygroup geometrygroup, RTvisibilitymask *mask)`
- `typedef RTResult(* PRTGEOMETRYSETFLAGS )(RTgeometry geometry, RTgeometryflags flags)`

- `typedef RTResult(* PRTGEOMETRYGETFLAGS )(RTgeometry geometry, RTgeometryflags *flags)`
- `typedef RTResult(* PRTMATERIALCREATE )(RTcontext context_api, RTmaterial *material)`
- `typedef RTResult(* PRTMATERIALDECLAREVARIABLE )(RTmaterial material_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTMATERIALDESTROY )(RTmaterial material_api)`
- `typedef RTResult(* PRTMATERIALGETANYHITPROGRAM )(RTmaterial material_api, unsigned int ray_type_index, RTprogram *program)`
- `typedef RTResult(* PRTMATERIALGETCLOSESTHITPROGRAM )(RTmaterial material_api, unsigned int ray_type_index, RTprogram *program)`
- `typedef RTResult(* PRTMATERIALGETCONTEXT )(RTmaterial material_api, RTcontext *c)`
- `typedef RTResult(* PRTMATERIALGETVARIABLE )(RTmaterial material_api, unsigned int index, RTvariable *v)`
- `typedef RTResult(* PRTMATERIALGETVARIABLECOUNT )(RTmaterial material_api, unsigned int *c)`
- `typedef RTResult(* PRTMATERIALQUERYVARIABLE )(RTmaterial material_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTMATERIALREMOVEVARIABLE )(RTmaterial material_api, RTvariable v_api)`
- `typedef RTResult(* PRTMATERIALSETANYHITPROGRAM )(RTmaterial material_api, unsigned int ray_type_index, RTprogram program_api)`
- `typedef RTResult(* PRTMATERIALSETCLOSESTHITPROGRAM )(RTmaterial material_api, unsigned int ray_type_index, RTprogram program_api)`
- `typedef RTResult(* PRTMATERIALVALIDATE )(RTmaterial material_api)`
- `typedef RTResult(* PRTPOSTPROCESSINGSTAGECREATEBUILTIN )(RTcontext context_api, const char *builtin_name, void *denoiser, void *ssim_predictor, RTpostprocessingstage *stage_api)`
- `typedef RTResult(* PRTPOSTPROCESSINGSTAGEDECLAREVARIABLE )(RTpostprocessingstage stage_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTPOSTPROCESSINGSTAGEDESTROY )(RTpostprocessingstage stage_api)`
- `typedef RTResult(* PRTPOSTPROCESSINGSTAGEGETCONTEXT )(RTpostprocessingstage stage_api, RTcontext *c)`
- `typedef RTResult(* PRTPOSTPROCESSINGSTAGEQUERYVARIABLE )(RTpostprocessingstage stage_api, const char *name, RTvariable *variable)`
- `typedef RTResult(* PRTPOSTPROCESSINGSTAGEGETVARIABLECOUNT )(RTpostprocessingstage stage_api, unsigned int *count)`
- `typedef RTResult(* PRTPOSTPROCESSINGSTAGEGETVARIABLE )(RTpostprocessingstage stage_api, unsigned int index, RTvariable *variable)`
- `typedef RTResult(* PRTPROGRAMCREATEFROMPTXFILE )(RTcontext context_api, const char *filename, const char *program_name, RTprogram *program)`
- `typedef RTResult(* PRTPROGRAMCREATEFROMPTXFILES )(RTcontext context_api, unsigned int n, const char **filenames, const char *program_name, RTprogram *program)`
- `typedef RTResult(* PRTPROGRAMCREATEFROMPTXSTRING )(RTcontext context_api, const char *ptx, const char *program_name, RTprogram *program)`
- `typedef RTResult(* PRTPROGRAMCREATEFROMPTXSTRINGS )(RTcontext context_api, unsigned int n, const char **ptx_strings, const char *program_name, RTprogram *program)`

- `typedef RTResult(* PRTPROGRAMDECLARE VARIABLE )(RTprogram program_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTPROGRAMDESTROY )(RTprogram program_api)`
- `typedef RTResult(* PRTPROGRAMGETCONTEXT )(RTprogram program_api, RTcontext *c)`
- `typedef RTResult(* PRTPROGRAMGETID )(RTprogram program_api, int *program_id)`
- `typedef RTResult(* PRTPROGRAMGETVARIABLE )(RTprogram program_api, unsigned int index, RTvariable *v)`
- `typedef RTResult(* PRTPROGRAMGETVARIABLECOUNT )(RTprogram program_api, unsigned int *c)`
- `typedef RTResult(* PRTPROGRAMQUERYVARIABLE )(RTprogram program_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTPROGRAMMOVEVARIABLE )(RTprogram program_api, RTvariable v_api)`
- `typedef RTResult(* PRTPROGRAMVALIDATE )(RTprogram program_api)`
- `typedef RTResult(* PRTPROGRAMCALLSITESETPOTENTIALCALLEES )(RTprogram program_api, const char *csName, const int *calleeIds, int numIds)`
- `typedef RTResult(* PRTSELECTORCREATE )(RTcontext context_api, RTselector *selector)`
- `typedef RTResult(* PRTSELECTORDECLARE VARIABLE )(RTselector selector_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTSELECTORDESTROY )(RTselector selector_api)`
- `typedef RTResult(* PRTSELECTORGETCHILD )(RTselector selector_api, unsigned int index, RTobject *child)`
- `typedef RTResult(* PRTSELECTORGETCHILDCOUNT )(RTselector selector_api, unsigned int *count)`
- `typedef RTResult(* PRTSELECTORGETCHILDTYPE )(RTselector selector_api, unsigned int index, RTobjecttype *type)`
- `typedef RTResult(* PRTSELECTORGETCONTEXT )(RTselector selector_api, RTcontext *c)`
- `typedef RTResult(* PRTSELECTORGETVARIABLE )(RTselector selector_api, unsigned int index, RTvariable *v)`
- `typedef RTResult(* PRTSELECTORGETVARIABLECOUNT )(RTselector selector_api, unsigned int *c)`
- `typedef RTResult(* PRTSELECTORGETVISITPROGRAM )(RTselector selector_api, RTprogram *program)`
- `typedef RTResult(* PRTSELECTORQUERYVARIABLE )(RTselector selector_api, const char *name, RTvariable *v)`
- `typedef RTResult(* PRTSELECTORMOVEVARIABLE )(RTselector selector_api, RTvariable v_api)`
- `typedef RTResult(* PRTSELECTORSETCHILD )(RTselector selector_api, unsigned int index, RTobject child)`
- `typedef RTResult(* PRTSELECTORSETCHILDCOUNT )(RTselector selector_api, unsigned int count)`
- `typedef RTResult(* PRTSELECTORSETVISITPROGRAM )(RTselector selector_api, RTprogram program_api)`
- `typedef RTResult(* PRTSELECTORVALIDATE )(RTselector selector_api)`
- `typedef RTResult(* PRTTEXTURESAMPLERCREATE )(RTcontext context_api, RTtexturesampler *textureSampler)`

- `typedef RTResult(* PRTTEXTURESAMPLERCREATEFROMGLIMAGE )(RTcontext context_api, unsigned int gl_id, RTgtarget target, RTtexturesampler *textureSampler)`
- `typedef RTResult(* PRTTEXTURESAMPLERDESTROY )(RTtexturesampler textureSampler_api)`
- `typedef RTResult(* PRTTEXTURESAMPLERGLREGISTER )(RTtexturesampler textureSampler_api)`
- `typedef RTResult(* PRTTEXTURESAMPLERGLUNREGISTER )(RTtexturesampler textureSampler_api)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETARRAYSIZE )(RTtexturesampler textureSampler_api, unsigned int *deprecated)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETBUFFER )(RTtexturesampler textureSampler_api, unsigned int deprecated0, unsigned int deprecated1, RTbuffer *buffer)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETCONTEXT )(RTtexturesampler textureSampler_api, RTcontext *c)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETFILTERINGMODES )(RTtexturesampler textureSampler_api, RTfiltermode *minFilter, RTfiltermode *magFilter, RTfiltermode *mipFilter)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETGLIMAGEID )(RTtexturesampler textureSampler_api, unsigned int *gl_id)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETID )(RTtexturesampler textureSampler_api, int *texture_id)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETINDEXINGMODE )(RTtexturesampler textureSampler_api, RTtextureindexmode *indexmode)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETMAXANISOTROPY )(RTtexturesampler textureSampler_api, float *maxAnisotropy)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETMIPLEVELCLAMP )(RTtexturesampler textureSampler_api, float *minLevel, float *maxLevel)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETMIPLEVELBIAS )(RTtexturesampler textureSampler_api, float *bias)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETMIPLEVELCOUNT )(RTtexturesampler textureSampler_api, unsigned int *deprecated)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETREADMODE )(RTtexturesampler textureSampler_api, RTtexturereadmode *readmode)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETWRAPMODE )(RTtexturesampler textureSampler_api, unsigned int dim, RTwrapmode *wm)`
- `typedef RTResult(* PRTTEXTURESAMPLERSETARRAYSIZE )(RTtexturesampler textureSampler_api, unsigned int deprecated)`
- `typedef RTResult(* PRTTEXTURESAMPLERSETBUFFER )(RTtexturesampler textureSampler_api, unsigned int deprecated0, unsigned int deprecated1, RTbuffer buffer_api)`
- `typedef RTResult(* PRTTEXTURESAMPLERSETFILTERINGMODES )(RTtexturesampler textureSampler_api, RTfiltermode minFilter, RTfiltermode magFilter, RTfiltermode mipFilter)`
- `typedef RTResult(* PRTTEXTURESAMPLERSETINDEXINGMODE )(RTtexturesampler textureSampler_api, RTtextureindexmode indexmode)`
- `typedef RTResult(* PRTTEXTURESAMPLERSETMAXANISOTROPY )(RTtexturesampler textureSampler_api, float maxAnisotropy)`
- `typedef RTResult(* PRTTEXTURESAMPLERSETMIPLEVELCLAMP )(RTtexturesampler textureSampler_api, float minLevel, float maxLevel)`
- `typedef RTResult(* PRTTEXTURESAMPLERSETMIPLEVELBIAS )(RTtexturesampler textureSampler_api, float bias)`

- `typedef RTResult(* PRTTEXTURESAMPLERSETMIPLEVELCOUNT )(RTtexturesampler textureSampler_api, unsigned int deprecated)`
- `typedef RTResult(* PRTTEXTURESAMPLERSETREADMODE )(RTtexturesampler textureSampler_api, RTtexturereadmode readmode)`
- `typedef RTResult(* PRTTEXTURESAMPLERSETWRAPMODE )(RTtexturesampler textureSampler_api, unsigned int dim, RTwrapmode wm)`
- `typedef RTResult(* PRTTEXTURESAMPLERVALIDATE )(RTtexturesampler textureSampler_api)`
- `typedef RTResult(* PRTTRANSFORMCREATE )(RTcontext context_api, RTtransform *transform)`
- `typedef RTResult(* PRTTRANSFORMDESTROY )(RTtransform transform_api)`
- `typedef RTResult(* PRTTRANSFORMGETCHILD )(RTtransform transform_api, RTobject *child)`
- `typedef RTResult(* PRTTRANSFORMGETCHILDTYPE )(RTtransform transform_api, RTobjecttype *type)`
- `typedef RTResult(* PRTTRANSFORMGETCONTEXT )(RTtransform transform_api, RTcontext *c)`
- `typedef RTResult(* PRTTRANSFORMGETMATRIX )(RTtransform transform_api, int transpose, float *matrix, float *inverse_matrix)`
- `typedef RTResult(* PRTTRANSFORMSETCHILD )(RTtransform transform_api, RTobject child)`
- `typedef RTResult(* PRTTRANSFORMSETMATRIX )(RTtransform transform_api, int transpose, const float *matrix, const float *inverse_matrix)`
- `typedef RTResult(* PRTTRANSFORMSETMOTIONRANGE )(RTtransform transform_api, float timeBegin, float timeEnd)`
- `typedef RTResult(* PRTTRANSFORMGETMOTIONRANGE )(RTtransform transform_api, float *timeBegin, float *timeEnd)`
- `typedef RTResult(* PRTTRANSFORMSETMOTIONBORDERMODE )(RTtransform transform_api, RTmotionbordermode beginMode, RTmotionbordermode endMode)`
- `typedef RTResult(* PRTTRANSFORMGETMOTIONBORDERMODE )(RTtransform transform_api, RTmotionbordermode *beginMode, RTmotionbordermode *endMode)`
- `typedef RTResult(* PRTTRANSFORMSETMOTIONKEYS )(RTtransform transform_api, unsigned int n, RTmotionkeytype type, const float *keys)`
- `typedef RTResult(* PRTTRANSFORMGETMOTIONKEYCOUNT )(RTtransform transform_api, unsigned int *n)`
- `typedef RTResult(* PRTTRANSFORMGETMOTIONKEYTYPE )(RTtransform transform_api, RTmotionkeytype *type)`
- `typedef RTResult(* PRTTRANSFORMGETMOTIONKEYS )(RTtransform transform_api, float *keys)`
- `typedef RTResult(* PRTTRANSFORMVALIDATE )(RTtransform transform_api)`
- `typedef RTResult(* PRTVARIABLEGET1F )(RTvariable v, float *f1)`
- `typedef RTResult(* PRTVARIABLEGET1FV )(RTvariable v, float *f)`
- `typedef RTResult(* PRTVARIABLEGET1I )(RTvariable v, int *i1)`
- `typedef RTResult(* PRTVARIABLEGET1IV )(RTvariable v, int *i)`
- `typedef RTResult(* PRTVARIABLEGET1UI )(RTvariable v, unsigned int *ull1)`
- `typedef RTResult(* PRTVARIABLEGET1UIV )(RTvariable v, unsigned int *ull)`
- `typedef RTResult(* PRTVARIABLEGET1LL )(RTvariable v, long long *ll1)`
- `typedef RTResult(* PRTVARIABLEGET1LLV )(RTvariable v, long long *ll)`
- `typedef RTResult(* PRTVARIABLEGET1ULL )(RTvariable v, unsigned long long *ull1)`
- `typedef RTResult(* PRTVARIABLEGET1ULLV )(RTvariable v, unsigned long long *ull)`
- `typedef RTResult(* PRTVARIABLEGET2F )(RTvariable v, float *f1, float *f2)`
- `typedef RTResult(* PRTVARIABLEGET2FV )(RTvariable v, float *f)`

- `typedef RTResult(* PRTVARIABLEGET2I )(RTvariable v, int *i1, int *i2)`
- `typedef RTResult(* PRTVARIABLEGET2IV )(RTvariable v, int *i)`
- `typedef RTResult(* PRTVARIABLEGET2UI )(RTvariable v, unsigned int *ull1, unsigned int *ull2)`
- `typedef RTResult(* PRTVARIABLEGET2UIV )(RTvariable v, unsigned int *ull)`
- `typedef RTResult(* PRTVARIABLEGET2LL )(RTvariable v, long long *ll1, long long *ll2)`
- `typedef RTResult(* PRTVARIABLEGET2LLV )(RTvariable v, long long *ll)`
- `typedef RTResult(* PRTVARIABLEGET2ULL )(RTvariable v, unsigned long long *ull1, unsigned long long *ull2)`
- `typedef RTResult(* PRTVARIABLEGET2ULLV )(RTvariable v, unsigned long long *ull)`
- `typedef RTResult(* PRTVARIABLEGET3F )(RTvariable v, float *f1, float *f2, float *f3)`
- `typedef RTResult(* PRTVARIABLEGET3FV )(RTvariable v, float *f)`
- `typedef RTResult(* PRTVARIABLEGET3I )(RTvariable v, int *i1, int *i2, int *i3)`
- `typedef RTResult(* PRTVARIABLEGET3IV )(RTvariable v, int *i)`
- `typedef RTResult(* PRTVARIABLEGET3UI )(RTvariable v, unsigned int *u1, unsigned int *u2, unsigned int *u3)`
- `typedef RTResult(* PRTVARIABLEGET3UIV )(RTvariable v, unsigned int *u)`
- `typedef RTResult(* PRTVARIABLEGET3LL )(RTvariable v, long long *ll1, long long *ll2, long long *ll3)`
- `typedef RTResult(* PRTVARIABLEGET3LLV )(RTvariable v, long long *ll)`
- `typedef RTResult(* PRTVARIABLEGET3ULL )(RTvariable v, unsigned long long *ull1, unsigned long long *ull2, unsigned long long *ull3)`
- `typedef RTResult(* PRTVARIABLEGET3ULLV )(RTvariable v, unsigned long long *ull)`
- `typedef RTResult(* PRTVARIABLEGET4F )(RTvariable v, float *f1, float *f2, float *f3, float *f4)`
- `typedef RTResult(* PRTVARIABLEGET4FV )(RTvariable v, float *f)`
- `typedef RTResult(* PRTVARIABLEGET4I )(RTvariable v, int *i1, int *i2, int *i3, int *i4)`
- `typedef RTResult(* PRTVARIABLEGET4IV )(RTvariable v, int *i)`
- `typedef RTResult(* PRTVARIABLEGET4UI )(RTvariable v, unsigned int *u1, unsigned int *u2, unsigned int *u3, unsigned int *u4)`
- `typedef RTResult(* PRTVARIABLEGET4UIV )(RTvariable v, unsigned int *u)`
- `typedef RTResult(* PRTVARIABLEGET4LL )(RTvariable v, long long *ll1, long long *ll2, long long *ll3, long long *ll4)`
- `typedef RTResult(* PRTVARIABLEGET4LLV )(RTvariable v, long long *ll)`
- `typedef RTResult(* PRTVARIABLEGET4ULL )(RTvariable v, unsigned long long *ull1, unsigned long long *ull2, unsigned long long *ull3, unsigned long long *ull4)`
- `typedef RTResult(* PRTVARIABLEGET4ULLV )(RTvariable v, unsigned long long *ull)`
- `typedef RTResult(* PRTVARIABLEGETANNOTATION )(RTvariable v, const char **annotation_return)`
- `typedef RTResult(* PRTVARIABLEGETCONTEXT )(RTvariable v, RTcontext *context)`
- `typedef RTResult(* PRTVARIABLEGETMATRIX2X2FV )(RTvariable v, int transpose, float *m)`
- `typedef RTResult(* PRTVARIABLEGETMATRIX2X3FV )(RTvariable v, int transpose, float *m)`
- `typedef RTResult(* PRTVARIABLEGETMATRIX2X4FV )(RTvariable v, int transpose, float *m)`
- `typedef RTResult(* PRTVARIABLEGETMATRIX3X2FV )(RTvariable v, int transpose, float *m)`
- `typedef RTResult(* PRTVARIABLEGETMATRIX3X3FV )(RTvariable v, int transpose, float *m)`
- `typedef RTResult(* PRTVARIABLEGETMATRIX3X4FV )(RTvariable v, int transpose, float *m)`
- `typedef RTResult(* PRTVARIABLEGETMATRIX4X2FV )(RTvariable v, int transpose, float *m)`
- `typedef RTResult(* PRTVARIABLEGETMATRIX4X3FV )(RTvariable v, int transpose, float *m)`

- `typedef RTResult(* PRTVARIABLEGETMATRIX4X4FV )(RTvariable v, int transpose, float *m)`
- `typedef RTResult(* PRTVARIABLEGETNAME )(RTvariable v, const char **name_return)`
- `typedef RTResult(* PRTVARIABLEGETOBJECT )(RTvariable v, RTobject *object)`
- `typedef RTResult(* PRTVARIABLEGETSIZE )(RTvariable v, RTsize *size)`
- `typedef RTResult(* PRTVARIABLEGETTYPE )(RTvariable v, RTobjecttype *type_return)`
- `typedef RTResult(* PRTVARIABLEGETUSERDATA )(RTvariable v, RTsize size, void *ptr)`
- `typedef RTResult(* PRTVARIABLESET1F )(RTvariable v, float f1)`
- `typedef RTResult(* PRTVARIABLESET1FV )(RTvariable v, const float *f)`
- `typedef RTResult(* PRTVARIABLESET1I )(RTvariable v, int i1)`
- `typedef RTResult(* PRTVARIABLESET1IV )(RTvariable v, const int *i)`
- `typedef RTResult(* PRTVARIABLESET1UI )(RTvariable v, unsigned int u1)`
- `typedef RTResult(* PRTVARIABLESET1UIV )(RTvariable v, const unsigned int *u)`
- `typedef RTResult(* PRTVARIABLESET1LL )(RTvariable v, long long i1)`
- `typedef RTResult(* PRTVARIABLESET1LLV )(RTvariable v, const long long *i)`
- `typedef RTResult(* PRTVARIABLESET1ULL )(RTvariable v, unsigned long long u1)`
- `typedef RTResult(* PRTVARIABLESET1ULLV )(RTvariable v, const unsigned long long *u)`
- `typedef RTResult(* PRTVARIABLESET2F )(RTvariable v, float f1, float f2)`
- `typedef RTResult(* PRTVARIABLESET2FV )(RTvariable v, const float *f)`
- `typedef RTResult(* PRTVARIABLESET2I )(RTvariable v, int i1, int i2)`
- `typedef RTResult(* PRTVARIABLESET2IV )(RTvariable v, const int *i)`
- `typedef RTResult(* PRTVARIABLESET2UI )(RTvariable v, unsigned int u1, unsigned int u2)`
- `typedef RTResult(* PRTVARIABLESET2UIV )(RTvariable v, const unsigned int *u)`
- `typedef RTResult(* PRTVARIABLESET2LL )(RTvariable v, long long i1, long long i2)`
- `typedef RTResult(* PRTVARIABLESET2LLV )(RTvariable v, const long long *i)`
- `typedef RTResult(* PRTVARIABLESET2ULL )(RTvariable v, unsigned long long u1, unsigned long long u2)`
- `typedef RTResult(* PRTVARIABLESET2ULLV )(RTvariable v, const unsigned long long *u)`
- `typedef RTResult(* PRTVARIABLESET3F )(RTvariable v, float f1, float f2, float f3)`
- `typedef RTResult(* PRTVARIABLESET3FV )(RTvariable v, const float *f)`
- `typedef RTResult(* PRTVARIABLESET3I )(RTvariable v, int i1, int i2, int i3)`
- `typedef RTResult(* PRTVARIABLESET3IV )(RTvariable v, const int *i)`
- `typedef RTResult(* PRTVARIABLESET3UI )(RTvariable v, unsigned int u1, unsigned int u2, unsigned int u3)`
- `typedef RTResult(* PRTVARIABLESET3UIV )(RTvariable v, const unsigned int *u)`
- `typedef RTResult(* PRTVARIABLESET3LL )(RTvariable v, long long i1, long long i2, long long i3)`
- `typedef RTResult(* PRTVARIABLESET3LLV )(RTvariable v, const long long *i)`
- `typedef RTResult(* PRTVARIABLESET3ULL )(RTvariable v, unsigned long long u1, unsigned long long u2, unsigned long long u3)`
- `typedef RTResult(* PRTVARIABLESET3ULLV )(RTvariable v, const unsigned long long *u)`
- `typedef RTResult(* PRTVARIABLESET4F )(RTvariable v, float f1, float f2, float f3, float f4)`
- `typedef RTResult(* PRTVARIABLESET4FV )(RTvariable v, const float *f)`
- `typedef RTResult(* PRTVARIABLESET4I )(RTvariable v, int i1, int i2, int i3, int i4)`
- `typedef RTResult(* PRTVARIABLESET4IV )(RTvariable v, const int *i)`
- `typedef RTResult(* PRTVARIABLESET4UI )(RTvariable v, unsigned int u1, unsigned int u2, unsigned int u3, unsigned int u4)`
- `typedef RTResult(* PRTVARIABLESET4UIV )(RTvariable v, const unsigned int *u)`

- `typedef RTResult(* PRTVARIABLESET4LL )(RTvariable v, long long i1, long long i2, long long i3, long long i4)`
- `typedef RTResult(* PRTVARIABLESET4LLV )(RTvariable v, const long long *i)`
- `typedef RTResult(* PRTVARIABLESET4ULL )(RTvariable v, unsigned long long u1, unsigned long long u2, unsigned long long u3, unsigned long long u4)`
- `typedef RTResult(* PRTVARIABLESET4ULLV )(RTvariable v, const unsigned long long *u)`
- `typedef RTResult(* PRTVARIABLESETMATRIX2X2FV )(RTvariable v, int transpose, const float *m)`
- `typedef RTResult(* PRTVARIABLESETMATRIX2X3FV )(RTvariable v, int transpose, const float *m)`
- `typedef RTResult(* PRTVARIABLESETMATRIX2X4FV )(RTvariable v, int transpose, const float *m)`
- `typedef RTResult(* PRTVARIABLESETMATRIX3X2FV )(RTvariable v, int transpose, const float *m)`
- `typedef RTResult(* PRTVARIABLESETMATRIX3X3FV )(RTvariable v, int transpose, const float *m)`
- `typedef RTResult(* PRTVARIABLESETMATRIX3X4FV )(RTvariable v, int transpose, const float *m)`
- `typedef RTResult(* PRTVARIABLESETMATRIX4X2FV )(RTvariable v, int transpose, const float *m)`
- `typedef RTResult(* PRTVARIABLESETMATRIX4X3FV )(RTvariable v, int transpose, const float *m)`
- `typedef RTResult(* PRTVARIABLESETMATRIX4X4FV )(RTvariable v, int transpose, const float *m)`
- `typedef RTResult(* PRTVARIABLESETOBJECT )(RTvariable v, RTobject object)`
- `typedef RTResult(* PRTVARIABLESETUSERDATA )(RTvariable v, RTsize size, const void *ptr)`
- `typedef RTResult(* PRTLOG )(int level, const char *msg)`
- `typedef RTResult(* PRTGETBUILDVERSION )(const char **result)`
- `typedef RTResult(* PRTOVERRIDESOTHERVERSION )(const char *otherVersion, int *result)`
- `typedef RTResult(* PRTDEPRECATEDAPIFUNCTION )()`
- `typedef struct IDirect3DDevice9 IDirect3DDevice9`
- `typedef struct IDirect3DResource9 IDirect3DResource9`
- `typedef struct IDXGIAdapter IDXGIAdapter`
- `typedef struct ID3D10Device ID3D10Device`
- `typedef struct ID3D10Resource ID3D10Resource`
- `typedef struct ID3D11Device ID3D11Device`
- `typedef struct ID3D11Resource ID3D11Resource`
- `typedef void * HGPUNV`
- `typedef RTResult(* PRTDEVICEGETWGLDEVICE )(int *device, HGPUNV hGpu)`
- `typedef RTResult(* PRTBUFFERCREATEFROMMD3D10RESOURCE )(RTcontext context_api, unsigned int type, ID3D10Resource *pResource, RTbuffer *buffer)`
- `typedef RTResult(* PRTBUFFERCREATEFROMMD3D11RESOURCE )(RTcontext context_api, unsigned int type, ID3D11Resource *pResource, RTbuffer *buffer)`
- `typedef RTResult(* PRTBUFFERCREATEFROMMD3D9RESOURCE )(RTcontext context_api, unsigned int type, IDirect3DResource9 *pResource, RTbuffer *buffer)`
- `typedef RTResult(* PRTBUFFERD3D10REGISTER )(RTbuffer buffer)`
- `typedef RTResult(* PRTBUFFERD3D10UNREGISTER )(RTbuffer buffer)`

- `typedef RTResult(* PRTBUFFERD3D11REGISTER )(RTbuffer buffer)`
- `typedef RTResult(* PRTBUFFERD3D11UNREGISTER )(RTbuffer buffer)`
- `typedef RTResult(* PRTBUFFERD3D9REGISTER )(RTbuffer buffer)`
- `typedef RTResult(* PRTBUFFERD3D9UNREGISTER )(RTbuffer buffer)`
- `typedef RTResult(* PRTBUFFERGETD3D10RESOURCE )(RTbuffer buffer_api, ID3D10Resource **pResource)`
- `typedef RTResult(* PRTBUFFERGETD3D11RESOURCE )(RTbuffer buffer_api, ID3D11Resource **pResource)`
- `typedef RTResult(* PRTBUFFERGETD3D9RESOURCE )(RTbuffer buffer_api, IDirect3DResource9 **pResource)`
- `typedef RTResult(* PRTCENTERSETD3D10DEVICE )(RTcontext context_api, ID3D10Device *matchingDevice)`
- `typedef RTResult(* PRTCENTERSETD3D11DEVICE )(RTcontext context_api, ID3D11Device *matchingDevice)`
- `typedef RTResult(* PRTCENTERSETD3D9DEVICE )(RTcontext context_api, IDirect3DDevice9 *matchingDevice)`
- `typedef RTResult(* PRTDEVICEGETD3D9DEVICE )(int *device, const char *pszAdapterName)`
- `typedef RTResult(* PRTDEVICEGETD3D10DEVICE )(int *device, IDXGIAdapter *pAdapter)`
- `typedef RTResult(* PRTDEVICEGETD3D11DEVICE )(int *device, IDXGIAdapter *pAdapter)`
- `typedef RTResult(* PRTTEXTURESAMPLERCREATEFROMD3D10RESOURCE )(RTcontext context_api, ID3D10Resource *pResource, RTtexturesampler *textureSampler)`
- `typedef RTResult(* PRTTEXTURESAMPLERCREATEFROMD3D11RESOURCE )(RTcontext context_api, ID3D11Resource *pResource, RTtexturesampler *textureSampler)`
- `typedef RTResult(* PRTTEXTURESAMPLERCREATEFROMD3D9RESOURCE )(RTcontext context_api, IDirect3DResource9 *pResource, RTtexturesampler *textureSampler)`
- `typedef RTResult(* PRTTEXTURESAMPLERD3D10REGISTER )(RTtexturesampler textureSampler)`
- `typedef RTResult(* PRTTEXTURESAMPLERD3D10UNREGISTER )(RTtexturesampler textureSampler)`
- `typedef RTResult(* PRTTEXTURESAMPLERD3D11REGISTER )(RTtexturesampler textureSampler)`
- `typedef RTResult(* PRTTEXTURESAMPLERD3D11UNREGISTER )(RTtexturesampler textureSampler)`
- `typedef RTResult(* PRTTEXTURESAMPLERD3D9REGISTER )(RTtexturesampler textureSampler)`
- `typedef RTResult(* PRTTEXTURESAMPLERD3D9UNREGISTER )(RTtexturesampler textureSampler)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETD3D10RESOURCE )(RTtexturesampler textureSampler_api, ID3D10Resource **pResource)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETD3D11RESOURCE )(RTtexturesampler textureSampler_api, ID3D11Resource **pResource)`
- `typedef RTResult(* PRTTEXTURESAMPLERGETD3D9RESOURCE )(RTtexturesampler textureSampler_api, IDirect3DResource9 **pResource)`
- `typedef struct OptiXAPI OptiXAPI_t`
- `typedef RTResult(* PRTGETSYMBOLTABLE )(unsigned int version, unsigned int numOptions, RTOptixoptions *options, void **option_values, OptiXAPI_t *symbol_table)`
- `typedef void *(* PRTDENOISERFACTORY )()`
- `typedef void *(* PRTSSIMPREDICTORFACTORY )()`

## Enumerations

- enum RTOptixoptions {  
    RT\_OPTIX\_OPTION\_FROM\_SDK = 0,  
    RT\_OPTIX\_OPTION\_SKIK\_DRIVER\_VERSION\_CHECK = 1 }



### 8.7.1 Macro Definition Documentation

8.7.1.1 **#define NVOPTIX\_ABI\_VERSION 12**

### 8.7.2 Typedef Documentation

8.7.2.1 **typedef void\* HGPUNV**

8.7.2.2 **typedef struct ID3D10Device ID3D10Device**

8.7.2.3 **typedef struct ID3D10Resource ID3D10Resource**

8.7.2.4 **typedef struct ID3D11Device ID3D11Device**

8.7.2.5 **typedef struct ID3D11Resource ID3D11Resource**

8.7.2.6 **typedef struct IDirect3DDevice9 IDirect3DDevice9**

8.7.2.7 **typedef struct IDirect3DResource9 IDirect3DResource9**

8.7.2.8 **typedef struct IDXGIAdapter IDXGIAdapter**

8.7.2.9 **typedef struct OptiXAPI OptiXAPI\_t**

8.7.2.10 **typedef RTResult( \* PRTACCELERATIONCREATE)(RTcontext context\_api,  
RTacceleration \*acceleration)**

8.7.2.11 **typedef RTResult( \* PRTACCELERATIONDESTROY)(RTacceleration acceleration\_api)**

8.7.2.12 **typedef RTResult( \* PRTACCELERATIONGETBUILDER)(RTacceleration  
acceleration\_api, const char \*\*return\_string)**

8.7.2.13 **typedef RTResult( \* PRTACCELERATIONGETCONTEXT)(RTacceleration  
acceleration\_api, RTcontext \*c)**

8.7.2.14 **typedef RTResult( \* PRTACCELERATIONGETDATA)(RTacceleration acceleration\_api,  
void \*data)**

8.7.2.15 **typedef RTResult( \* PRTACCELERATIONGETDATASIZE)(RTacceleration  
acceleration\_api, RTsize \*size)**

8.7.2.16 **typedef RTResult( \* PRTACCELERATIONGETPROPERTY)(RTacceleration  
acceleration\_api, const char \*name, const char \*\*return\_string)**

8.7.2.17 **typedef RTResult( \* PRTACCELERATIONGETTRAVERSER)(RTacceleration  
acceleration\_api, const char \*\*return\_string)**

8.7.2.18 **typedef RTResult( \* PRTACCELERATIONISDIRTY)(RTacceleration acceleration\_api, int  
\*dirty)**

---

8.7.2.19 **typedef RTResult( \* PRTACCELERATIONMARKDIRTY)(RTacceleration acceleration\_api)**

8.7.2.20 **typedef RTResult( \* PRTACCELERATIONSETBUILDER)(RTacceleration  
acceleration\_api, const char \*builder)**

name of the rtcore DLL. Option value: 1 bool

***RT\_OPTIX\_OPTION\_SKIK\_DRIVER\_VERSION\_CHECK*** Flag that indicates whether `rtGetSymbolTable()` should skip the check for the minimum required driver version.

## 8.8 optix.h File Reference

### Macros

- `#define OPTIX_VERSION`

#### 8.8.1 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`

#### 8.8.2 Macro Definition Documentation

##### 8.8.2.1 `#define OPTIX_VERSION`

**Value:**

```
60000 /* major = OPTIX_VERSION/10000, *
 * minor = (OPTIX_VERSION%10000)/100, *
 * micro = OPTIX_VERSION%100 */
```

## 8.9 optix\_cuda.h File Reference

## 8.10 optix\_cuda\_interop.h File Reference

### 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)`

### 8.10.1 Detailed Description

OptiX public API declarations CUDAInterop.

Author

NVIDIA Corporation OptiX public API declarations for CUDA interoperability

## 8.11 optix\_datatypes.h File Reference

### 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)

### 8.11.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Datatypes

### 8.11.2 Macro Definition Documentation

#### 8.11.2.1 #define RT\_DEFAULT\_MAX 1.e27f

Max t for a ray.

### 8.11.3 Function Documentation

#### 8.11.3.1 static \_\_inline\_\_ RT\_HOSTDEVICE Ray make\_Ray (

```
 float3 origin,
 float3 direction,
 unsigned int ray_type,
 float tmin,
```

```
float tmax) [static]
```

## 8.12 optix\_declarations.h File Reference

### Macros

- `#define RT_HOSTDEVICE`

### Typedefs

- `typedef unsigned int RTvisibilitymask`

### 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,
RT_FORMAT_LONG_LONG,
RT_FORMAT_LONG_LONG2,
RT_FORMAT_LONG_LONG3,
RT_FORMAT_LONG_LONG4,
RT_FORMAT_UNSIGNED_LONG_LONG,
RT_FORMAT_UNSIGNED_LONG_LONG2,
RT_FORMAT_UNSIGNED_LONG_LONG3,
RT_FORMAT_UNSIGNED_LONG_LONG4,
RT_FORMAT_UNSIGNED_BC1,
RT_FORMAT_UNSIGNED_BC2,
RT_FORMAT_UNSIGNED_BC3,
RT_FORMAT_UNSIGNED_BC4,
RT_FORMAT_BC4,
RT_FORMAT_UNSIGNED_BC5,
RT_FORMAT_BC5,
RT_FORMAT_UNSIGNED_BC6H,
RT_FORMAT_BC6H,
RT_FORMAT_UNSIGNED_BC7 }

• 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,
RT_OBJECTTYPE_COMMANDLIST,
RT_OBJECTTYPE_POSTPROCESSINGSTAGE,
RT_OBJECTTYPE_LONG_LONG,
RT_OBJECTTYPE_LONG_LONG2,
RT_OBJECTTYPE_LONG_LONG3,
RT_OBJECTTYPE_LONG_LONG4,
RT_OBJECTTYPE_UNSIGNED_LONG_LONG,
RT_OBJECTTYPE_UNSIGNED_LONG_LONG2,
RT_OBJECTTYPE_UNSIGNED_LONG_LONG3,
RT_OBJECTTYPE_UNSIGNED_LONG_LONG4 }

• 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_DISCARD_HOST_MEMORY = 0x20,
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_PAYLOAD_ACCESS_OUT_OF_BOUNDS = 0x3EB,
 RT_EXCEPTION_USER_EXCEPTION_CODE_OUT_OF_BOUNDS = 0x3EC,
 RT_EXCEPTION_TRACE_DEPTH_EXCEEDED = 0x3ED,
 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_USER_MAX = 0xFFFF,
 RT_EXCEPTION_ALL = 0x7FFFFFFF }

• enum RTresult {
 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_OPTIX_NOT_LOADED = 0x60A,
 RT_ERROR_DENOISER_NOT_LOADED = 0x60B,
 RT_ERROR_SSIM_PREDICTOR_NOT_LOADED = 0x60C,
```

```
RT_ERROR_DRIVER_VERSION_FAILED = 0x60D,
RT_ERROR_DATABASE_FILE_PERMISSIONS = 0x60E,
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_INVALID_GLOBAL_ATTRIBUTE = 0xC00,
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,
 RT_DEVICE_ATTRIBUTE_PCI_BUS_ID,
 RT_DEVICE_ATTRIBUTE_COMPATIBLE_DEVICES }

• enum RTglobalattribute {
 RT_GLOBAL_ATTRIBUTE_DISPLAY_DRIVER_VERSION_MAJOR =1,
 RT_GLOBAL_ATTRIBUTE_DISPLAY_DRIVER_VERSION_MINOR,
 RT_GLOBAL_ATTRIBUTE_ENABLE_RTX = 0x10000000,
 RT_GLOBAL_ATTRIBUTE_DEVELOPER_OPTIONS }

• 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_DISK_CACHE_ENABLED,
 RT_CONTEXT_ATTRIBUTE_PREFER_FAST_RECOMPILES,
 RT_CONTEXT_ATTRIBUTE_FORCE_INLINE_USER_FUNCTIONS,
 RT_CONTEXT_ATTRIBUTE_OPTIX_SALT,
 RT_CONTEXT_ATTRIBUTE_VENDOR_SALT,
 RT_CONTEXT_ATTRIBUTE_PUBLIC_VENDOR_KEY,
 RT_CONTEXT_ATTRIBUTE_DISK_CACHE_LOCATION,
 RT_CONTEXT_ATTRIBUTE_DISK_CACHE_MEMORY_LIMITS,
 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 RTmotionbordermode {
 RT_MOTIONBORDERMODE_CLAMP,
 RT_MOTIONBORDERMODE_VANISH }

• enum RTmotionkeytype {
 RT_MOTIONKEYTYPE_NONE = 0,
 RT_MOTIONKEYTYPE_MATRIX_FLOAT12,
 RT_MOTIONKEYTYPE_SRT_FLOAT16 }

• enum RTgeometrybuildflags {
 RT_GEOMETRY_BUILD_FLAG_NONE = 0x00,
 RT_GEOMETRY_BUILD_FLAG_RELEASE_BUFFERS = 0x10 }

• enum RTgeometryflags {
 RT_GEOMETRY_FLAG_NONE = 0x00,
 RT_GEOMETRY_FLAG_DISABLE_ANYHIT = 0x01,
 RT_GEOMETRY_FLAG_NO_SPLITTING = 0x02 }

• enum RTinstanceflags {
 RT_INSTANCE_FLAG_NONE = 0u,
 RT_INSTANCE_FLAG_DISABLE_TRIANGLE_CULLING = 1u << 0,
 RT_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u << 1,
 RT_INSTANCE_FLAG_DISABLE_ANYHIT = 1u << 2,
 RT_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u << 3 }

• enum RTbufferidnull { RT_BUFFER_ID_NULL = 0 }

• enum RTprogramidnull { RT_PROGRAM_ID_NULL = 0 }

• enum RTtextureidnull { RT_TEXTURE_ID_NULL = 0 }

• enum RTcommandlistidnull { RT_COMMAND_LIST_ID_NULL = 0 }

• enum RTpostprocessingstagenull { RT_POSTPROCESSING_STAGE_ID_NULL = 0 }

• enum RTrayflags {
 RT_RAY_FLAG_NONE = 0u,
 RT_RAY_FLAG_DISABLE_ANYHIT = 1u << 0,
 RT_RAY_FLAG_ENFORCE_ANYHIT = 1u << 1,
 RT_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u << 2,
 RT_RAY_FLAG_DISABLE_CLOSESTHIT = 1u << 3,
 RT_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u << 4,
 RT_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u << 5,
 RT_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u << 6,
 RT_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u << 7 }

• enum { RT_VISIBILITY_ALL = 0xFFu }
```

### 8.12.1 Detailed Description

OptiX public API declarations.

Author

NVIDIA Corporation OptiX public API declarations

## 8.12.2 Macro Definition Documentation

### 8.12.2.1 #define RT\_HOSTDEVICE

### 8.12.3 Typedef Documentation

#### 8.12.3.1 `typedef unsigned int RTvisibilitymask`

### 8.12.4 Enumeration Type Documentation

#### 8.12.4.1 anonymous enum

Enumerator

***RT\_VISIBILITY\_ALL*** Default `RTvisibilitymask`.

#### 8.12.4.2 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)`

#### 8.12.4.3 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\_DISCARD\_HOST\_MEMORY*** An `RT_BUFFER_INPUT` for which a synchronize is forced on unmapping from host and the host memory is freed.

***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

#### 8.12.4.4 enum RTbufferidnull

Sentinel values.

Enumerator

***RT\_BUFFER\_ID\_NULL*** sentinel for describing a non-existent buffer id

### 8.12.4.5 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.

### 8.12.4.6 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*** Ouput/Input buffer for the GPU.
- RT\_BUFFER\_PROGRESSIVE\_STREAM*** Progressive stream buffer.

### 8.12.4.7 enum RTcommandlistidnull

Enumerator

- RT\_COMMAND\_LIST\_ID\_NULL*** sentinel for describing a non-existent command list id

### 8.12.4.8 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\_DISK\_CACHE\_ENABLED*** sizeof(int)
- RT\_CONTEXT\_ATTRIBUTE\_PREFER\_FAST\_RECOMPILES*** sizeof(int)
- RT\_CONTEXT\_ATTRIBUTE\_FORCE\_INLINE\_USER\_FUNCTIONS*** sizeof(int)
- RT\_CONTEXT\_ATTRIBUTE\_OPTIX\_SALT*** 32
- RT\_CONTEXT\_ATTRIBUTE\_VENDOR\_SALT*** 32
- RT\_CONTEXT\_ATTRIBUTE\_PUBLIC\_VENDOR\_KEY*** variable
- RT\_CONTEXT\_ATTRIBUTE\_DISK\_CACHE\_LOCATION*** sizeof(char\*)
- RT\_CONTEXT\_ATTRIBUTE\_DISK\_CACHE\_MEMORY\_LIMITS*** sizeof(RTsize[2])
- RT\_CONTEXT\_ATTRIBUTE\_AVAILABLE\_DEVICE\_MEMORY*** sizeof(RTsize)

### 8.12.4.9 enum RTdeviceattribute

Device attributes.

Enumerator

***RT\_DEVICE\_ATTRIBUTE\_MAX\_THREADS\_PER\_BLOCK*** Max Threads per Block sizeof(int)

***RT\_DEVICE\_ATTRIBUTE\_CLOCK\_RATE*** Clock rate sizeof(int)

***RT\_DEVICE\_ATTRIBUTE\_MULTIPROCESSOR\_COUNT*** Multiprocessor count sizeof(int)

***RT\_DEVICE\_ATTRIBUTE\_EXECUTION\_TIMEOUT\_ENABLED*** Execution timeout enabled sizeof(int)

***RT\_DEVICE\_ATTRIBUTE\_MAX\_HARDWARE\_TEXTURE\_COUNT*** Hardware Texture count sizeof(int)

***RT\_DEVICE\_ATTRIBUTE\_NAME*** Attribute Name.

***RT\_DEVICE\_ATTRIBUTE\_COMPUTE\_CAPABILITY*** Compute Capabilities sizeof(int2)

***RT\_DEVICE\_ATTRIBUTE\_TOTAL\_MEMORY*** Total Memory sizeof(RTsize)

***RT\_DEVICE\_ATTRIBUTE\_TCC\_DRIVER*** TCC driver sizeof(int)

***RT\_DEVICE\_ATTRIBUTE\_CUDA\_DEVICE\_ORDINAL*** CUDA device ordinal sizeof(int)

***RT\_DEVICE\_ATTRIBUTE\_PCI\_BUS\_ID*** PCI Bus Id.

***RT\_DEVICE\_ATTRIBUTE\_COMPATIBLE\_DEVICES*** Ordinals of compatible devices sizeof(int=N) + N\*sizeof(int)

### 8.12.4.10 enum RTexception

Exceptions.

**See also** [rtContextSetExceptionEnabled](#), [rtContextGetExceptionEnabled](#), [rtGetExceptionCode](#), [rtThrow](#), [rtPrintf](#) functions

Enumerator

***RT\_EXCEPTION\_PAYLOAD\_ACCESS\_OUT\_OF\_BOUNDS*** Payload access out of bounds - Added in OptiX 6.0.

***RT\_EXCEPTION\_USER\_EXCEPTION\_CODE\_OUT\_OF\_BOUNDS*** Exception code of user exception out of bounds - Added in OptiX 6.0.

***RT\_EXCEPTION\_TRACE\_DEPTH\_EXCEEDED*** Trace depth exceeded - Added in Optix 6.0.

***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*** First user exception code.

***RT\_EXCEPTION\_USER\_MAX*** Last user exception code.

***RT\_EXCEPTION\_ALL*** All exceptions.

#### 8.12.4.11 enum RTfiltermode

Filter mode.

Enumerator

***RT\_FILTER\_NEAREST*** Nearest.

***RT\_FILTER\_LINEAR*** Linear.

***RT\_FILTER\_NONE*** No filter.

#### 8.12.4.12 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  
***RT\_FORMAT\_LONG\_LONG*** LONG\_LONG.  
***RT\_FORMAT\_LONG\_LONG2*** sizeof(LONG\_LONG)\*2  
***RT\_FORMAT\_LONG\_LONG3*** sizeof(LONG\_LONG)\*3  
***RT\_FORMAT\_LONG\_LONG4*** sizeof(LONG\_LONG)\*4  
***RT\_FORMAT\_UNSIGNED\_LONG\_LONG*** sizeof(ULONG\_LONG)  
***RT\_FORMAT\_UNSIGNED\_LONG\_LONG2*** sizeof(ULONG\_LONG)\*2  
***RT\_FORMAT\_UNSIGNED\_LONG\_LONG3*** sizeof(ULONG\_LONG)\*3  
***RT\_FORMAT\_UNSIGNED\_LONG\_LONG4*** sizeof(ULONG\_LONG)\*4  
***RT\_FORMAT\_UNSIGNED\_BC1*** Block Compressed RGB + optional 1-bit alpha BC1,  
  sizeof(UINT)\*2.  
***RT\_FORMAT\_UNSIGNED\_BC2*** Block Compressed RGB + 4-bit alpha BC2, sizeof(UINT)\*4.  
***RT\_FORMAT\_UNSIGNED\_BC3*** Block Compressed RGBA BC3, sizeof(UINT)\*4.  
***RT\_FORMAT\_UNSIGNED\_BC4*** Block Compressed unsigned grayscale BC4, sizeof(UINT)\*2.  
***RT\_FORMAT\_BC4*** Block Compressed signed grayscale BC4, sizeof(UINT)\*2.  
***RT\_FORMAT\_UNSIGNED\_BC5*** Block Compressed unsigned 2 x grayscale BC5,  
  sizeof(UINT)\*4.  
***RT\_FORMAT\_BC5*** Block compressed signed 2 x grayscale BC5, sizeof(UINT)\*4.  
***RT\_FORMAT\_UNSIGNED\_BC6H*** Block compressed BC6 unsigned half-float, sizeof(UINT)\*4.  
***RT\_FORMAT\_BC6H*** Block compressed BC6 signed half-float, sizeof(UINT)\*4.  
***RT\_FORMAT\_UNSIGNED\_BC7*** Block compressed BC7, sizeof(UINT)\*4.

#### 8.12.4.13 enum RTgeometrybuildflags

GeometryX build flags.

Enumerator

***RT\_GEOMETRY\_BUILD\_FLAG\_NONE*** No special flags set.  
***RT\_GEOMETRY\_BUILD\_FLAG\_RELEASE\_BUFFERS*** User buffers are released after  
  consumption by acceleration structure build.

#### 8.12.4.14 enum RTgeometryflags

Material-dependent flags set on Geometry/GeometryTriangles.

Enumerator

- RT\_GEOMETRY\_FLAG\_NONE*** No special flags set.
- RT\_GEOMETRY\_FLAG\_DISABLE\_ANYHIT*** Opaque flag, any hit program will be skipped.
- RT\_GEOMETRY\_FLAG\_NO\_SPLITTING*** Disable primitive splitting to avoid potential duplicate any hit program execution for a single intersection.

#### 8.12.4.15 enum RTglobalattribute

Global attributes.

Enumerator

- RT\_GLOBAL\_ATTRIBUTE\_DISPLAY\_DRIVER\_VERSION\_MAJOR*** sizeof(int)
- RT\_GLOBAL\_ATTRIBUTE\_DISPLAY\_DRIVER\_VERSION\_MINOR*** sizeof(int)
- RT\_GLOBAL\_ATTRIBUTE\_ENABLE\_RTX*** sizeof(int)
- RT\_GLOBAL\_ATTRIBUTE\_DEVELOPER\_OPTIONS*** Knobs string.

#### 8.12.4.16 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.

#### 8.12.4.17 enum RTinstanceflags

Instance flags which override the behavior of geometry.

Enumerator

- RT\_INSTANCE\_FLAG\_NONE*** No special flag set.
- RT\_INSTANCE\_FLAG\_DISABLE\_TRIANGLE\_CULLING*** Prevent triangles from getting culled.
- RT\_INSTANCE\_FLAG\_FLIP\_TRIANGLE\_FACING*** Flip triangle orientation. This affects front/backface culling.

***RT\_INSTANCE\_FLAG\_DISABLE\_ANYHIT*** Disable any-hit programs. This may yield significantly higher performance even in cases where no any-hit programs are set.

***RT\_INSTANCE\_FLAG\_ENFORCE\_ANYHIT*** Override  
***RT\_GEOMETRY\_FLAG\_DISABLE\_ANYHIT***.

#### 8.12.4.18 enum RTmotionbordermode

Motion border modes.

Enumerator

***RT\_MOTIONBORDERMODE\_CLAMP*** Clamp outside of bounds.

***RT\_MOTIONBORDERMODE\_VANISH*** Vanish outside of bounds.

#### 8.12.4.19 enum RTmotionkeytype

Motion key type.

Enumerator

***RT\_MOTIONKEYTYPE\_NONE*** No motion keys set.

***RT\_MOTIONKEYTYPE\_MATRIX\_FLOAT12*** Affine matrix format - 12 floats.

***RT\_MOTIONKEYTYPE\_SRT\_FLOAT16*** SRT format - 16 floats.

#### 8.12.4.20 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*** 32 Bit Integer Type

***RT\_OBJECTTYPE\_INT2*** 32 Bit Integer2 Type

***RT\_OBJECTTYPE\_INT3*** 32 Bit Integer3 Type

***RT\_OBJECTTYPE\_INT4*** 32 Bit Integer4 Type

***RT\_OBJECTTYPE\_UNSIGNED\_INT*** 32 Bit Unsigned Integer Type

***RT\_OBJECTTYPE\_UNSIGNED\_INT2*** 32 Bit Unsigned Integer2 Type

***RT\_OBJECTTYPE\_UNSIGNED\_INT3*** 32 Bit Unsigned Integer3 Type

***RT\_OBJECTTYPE\_UNSIGNED\_INT4*** 32 Bit Unsigned Integer4 Type

***RT\_OBJECTTYPE\_USER*** User Object Type.

***RT\_OBJECTTYPE\_PROGRAM*** Object Type Program - Added in OptiX 3.0.

***RT\_OBJECTTYPE\_COMMANDLIST*** Object Type Command List - Added in OptiX 5.0.

***RT\_OBJECTTYPE\_POSTPROCESSINGSTAGE*** Object Type Postprocessing Stage - Added in OptiX 5.0.

***RT\_OBJECTTYPE\_LONG\_LONG*** 64 Bit Integer Type - Added in Optix 6.0

***RT\_OBJECTTYPE\_LONG\_LONG2*** 64 Bit Integer2 Type - Added in Optix 6.0

***RT\_OBJECTTYPE\_LONG\_LONG3*** 64 Bit Integer3 Type - Added in Optix 6.0

***RT\_OBJECTTYPE\_LONG\_LONG4*** 64 Bit Integer4 Type - Added in Optix 6.0

***RT\_OBJECTTYPE\_UNSIGNED\_LONG\_LONG*** 64 Bit Unsigned Integer Type - Added in Optix 6.0

***RT\_OBJECTTYPE\_UNSIGNED\_LONG\_LONG2*** 64 Bit Unsigned Integer2 Type - Added in Optix 6.0

***RT\_OBJECTTYPE\_UNSIGNED\_LONG\_LONG3*** 64 Bit Unsigned Integer3 Type - Added in Optix 6.0

***RT\_OBJECTTYPE\_UNSIGNED\_LONG\_LONG4*** 64 Bit Unsigned Integer4 Type - Added in Optix 6.0

#### 8.12.4.21 enum RTpostprocessingstagenull

Enumerator

***RT\_POSTPROCESSING\_STAGE\_ID\_NULL*** sentinel for describing a non-existent post-processing stage id

### 8.12.4.22 enum RTprogramidnull

Enumerator

***RT\_PROGRAM\_ID\_NULL*** sentinel for describing a non-existent program id

### 8.12.4.23 enum RTrayflags

Ray flags.

Enumerator

***RT\_RAY\_FLAG\_NONE***

***RT\_RAY\_FLAG\_DISABLE\_ANYHIT*** Disables any-hit programs for the ray.

***RT\_RAY\_FLAG\_ENFORCE\_ANYHIT*** Forces any-hit program execution for the ray.

***RT\_RAY\_FLAG\_TERMINATE\_ON\_FIRST\_HIT*** Terminates the ray after the first hit.

***RT\_RAY\_FLAG\_DISABLE\_CLOSESTHIT*** Disables closest-hit programs for the ray.

***RT\_RAY\_FLAG\_CULL\_BACK\_FACING\_TRIANGLES*** Do not intersect triangle back faces.

***RT\_RAY\_FLAG\_CULL\_FRONT\_FACING\_TRIANGLES*** Do not intersect triangle front faces.

***RT\_RAY\_FLAG\_CULL\_DISABLED\_ANYHIT*** Do not intersect geometry which disables any-hit programs.

***RT\_RAY\_FLAG\_CULL\_ENFORCED\_ANYHIT*** Do not intersect geometry which enforces any-hit programs.

### 8.12.4.24 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\_OPTIX\_NOT\_LOADED*** OptiX DLL failed to load.

***RT\_ERROR\_DENOISER\_NOT\_LOADED*** Denoiser DLL failed to load.

***RT\_ERROR\_SSIM\_PREDICTOR\_NOT\_LOADED*** SSIM predictor DLL failed to load.

***RT\_ERROR\_DRIVER\_VERSION\_FAILED*** Driver version retrieval failed.

***RT\_ERROR\_DATABASE\_FILE\_PERMISSIONS*** No write permission on disk cache file.

***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\_INVALID\_GLOBAL\_ATTRIBUTE*** Invalid global attribute.

***RT\_ERROR\_UNKNOWN*** Error unknown.

#### 8.12.4.25 enum RTtextureidnull

Enumerator

***RT\_TEXTURE\_ID\_NULL*** sentinel for describing a non-existent texture id

#### 8.12.4.26 enum RTtextureindexmode

Texture index mode.

Enumerator

***RT\_TEXTURE\_INDEX\_NORMALIZED\_COORDINATES*** Texture Index normalized coordinates.

***RT\_TEXTURE\_INDEX\_ARRAY\_INDEX*** Texture Index Array.

### 8.12.4.27 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.

### 8.12.4.28 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.

## 8.13 optix\_declarations\_private.h File Reference

### Namespaces

- [optix](#)

### Constant Groups

- [optix](#)

### Macros

- `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_SUBFRAME_INDEX 0x2000006 /* sizeof(int) */`
- `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_RESERVE_PROGRAM_ID 0x2000010 /* sizeof(int) */`
- `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_RESERVE_BUFFER_ID 0x2000011 /* sizeof(int) */`
- `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_RESERVE_TEXTURE_SAMPLER_ID 0x2000012 /* sizeof(int) */`

- `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_PROGRAM_ID_HINT 0x2000013 /* sizeof(int) */`
- `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_BUFFER_ID_HINT 0x2000014 /* sizeof(int) */`
- `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_TEXTURE_SAMPLER_ID_HINT 0x2000015 /* sizeof(int) */`

## Enumerations

- enum `RTbufferflag_internal {`  
`RT_BUFFER_PARTITIONED_INTERNAL = 1u << 16,`  
`RT_BUFFER_PINNED_INTERNAL = 1u << 17,`  
`RT_BUFFER_WRITECOMBINED_INTERNAL = 1u << 18,`  
`RT_BUFFER_DEVICE_ONLY_INTERNAL = 1u << 19,`  
`RT_BUFFER_FORCE_ZERO_COPY = 1u << 20,`  
`RT_BUFFER_LAYERED_RESERVED = 1u << 21,`  
`RT_BUFFER_CUBEMAP_RESERVED = 1u << 22,`  
`RT_BUFFER_INTERNAL_PREFER_TEX_HEAP = 1u << 23,`  
`RT_BUFFER_HINT_STATIC = 1u << 24 }`
- enum `optix::ObjectStorageType {`  
`optix::OBJECT_STORAGE_CONSTANT,`  
`optix::OBJECT_STORAGE_SHARED,`  
`optix::OBJECT_STORAGE_GLOBAL,`  
`optix::OBJECT_STORAGE_LINEAR_TEXTURE,`  
`optix::OBJECT_STORAGE_BLOCKED_TEXTURE }`

### 8.13.1 Macro Definition Documentation

- 8.13.1.1 `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_BUFFER_ID_HINT 0x2000014 /*  
  sizeof(int) */`
- 8.13.1.2 `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_PROGRAM_ID_HINT 0x2000013 /*  
  sizeof(int) */`
- 8.13.1.3 `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_RESERVE_BUFFER_ID 0x2000011 /*  
  sizeof(int) */`
- 8.13.1.4 `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_RESERVE_PROGRAM_ID 0x2000010  
  /* sizeof(int) */`
- 8.13.1.5 `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_RESERVE_TEXTURE_SAMPLER_ID 0x2000012  
  /* sizeof(int) */`
- 8.13.1.6 `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_SUBFRAME_INDEX 0x2000006 /*  
  sizeof(int) */`
- 8.13.1.7 `#define RT_CONTEXT_INTERNAL_ATTRIBUTE_TEXTURE_SAMPLER_ID_HINT 0x2000015  
  /* sizeof(int) */`

### 8.13.2 Enumeration Type Documentation

#### 8.13.2.1 enum RTbufferflag\_internal

Enumerator

*RT\_BUFFER\_PARTITIONED\_INTERNAL*  
*RT\_BUFFER\_PINNED\_INTERNAL*  
*RT\_BUFFER\_WRITECOMBINED\_INTERNAL*  
*RT\_BUFFER\_DEVICE\_ONLY\_INTERNAL*  
*RT\_BUFFER\_FORCE\_ZERO\_COPY*  
*RT\_BUFFER\_LAYERED\_RESERVED*  
*RT\_BUFFER\_CUBEMAP\_RESERVED*  
*RT\_BUFFER\_INTERNAL\_PREFER\_TEX\_HEAP*  
*RT\_BUFFER\_HINT\_STATIC*

---

## 8.14 optixDefines.h File Reference

### Classes

- struct `rti_internal_typeinfo::rti_typeinfo`
- struct `rti_internal_typeinfo::rti_typeenum< T >`

## Namespaces

- rti\_internal\_typeinfo
- optix

## Constant Groups

- rti\_internal\_typeinfo
- optix

## 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 rti\_internal\_typeinfo::rtiTypeKind { rti\_internal\_typeinfo::\_OPTIX\_VARIABLE = 0x796152 }
- enum rti\_internal\_typeinfo::rtiTypeEnum {  
    rti\_internal\_typeinfo::\_OPTIX\_TYPE\_ENUM\_UNKNOWN = 0x1337,  
    rti\_internal\_typeinfo::\_OPTIX\_TYPE\_ENUM\_PROGRAM\_ID,  
    rti\_internal\_typeinfo::\_OPTIX\_TYPE\_ENUM\_PROGRAM\_AS\_ID }
- enum optix::rtiTTexLookupKind {  
    optix::TEX\_LOOKUP\_1D = 1,  
    optix::TEX\_LOOKUP\_2D = 2,  
    optix::TEX\_LOOKUP\_3D = 3,  
    optix::TEX\_LOOKUP\_A1 = 4,  
    optix::TEX\_LOOKUP\_A2 = 5,  
    optix::TEX\_LOOKUP\_CUBE = 6,  
    optix::TEX\_LOOKUP\_ACUBE = 7 }

### 8.14.1 Detailed Description

OptiX public API.

## Author

NVIDIA Corporation OptiX public API Reference - Definitions

### 8.14.2 Macro Definition Documentation

**8.14.2.1 #define OPTIX\_ASM\_PTR "r"**

**8.14.2.2 #define OPTIX\_ASM\_PTR\_SIZE\_STR "32"**

**8.14.2.3 #define OPTIX\_ASM\_SIZE\_T "r"**

**8.14.2.4 #define OPTIX\_BITNESS\_SUFFIX ""**

### 8.14.3 Enumeration Type Documentation

**8.14.3.1 enum RTtransformflags**

Transform flags.

Enumerator

***RT\_INTERNAL\_INVERSE\_TRANSPOSE*** Inverse transpose flag.

**8.14.3.2 enum RTtransformkind**

Transform type.

Enumerator

***RT\_WORLD\_TO\_OBJECT*** World to Object transformation.

***RT\_OBJECT\_TO\_WORLD*** Object to World transformation.

## 8.15 optix\_device.h File Reference

### 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\_CPAvgVoid< T1 >
- struct rti\_internal\_callableprogram::is\_CPAvgVoid< CPArgVoid >
- struct rti\_internal\_callableprogram::check\_is\_CPAvgVoid< Condition, Dummy >
- struct rti\_internal\_callableprogram::check\_is\_CPAvgVoid< false, IntentionalError >
- class rti\_internal\_callableprogram::callableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T >
- class rti\_internal\_callableprogram::markedCallableProgramIdBase< ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T >
- class optix::callableProgramId< T >
- class optix::markedCallableProgramId< T >
- class optix::boundCallableProgramId< T >
- struct rti\_internal\_typeinfo::rti\_typeenum< optix::callableProgramId< T > >
- struct rti\_internal\_typeinfo::rti\_typeenum< optix::boundCallableProgramId< T > >

## Namespaces

- optix
- rti\_internal\_callableprogram
- rti\_internal\_typeinfo

## Constant Groups

- optix
- rti\_internal\_callableprogram
- rti\_internal\_typeinfo

## Macros

- #define rtDeclareVariable(type, name, semantic, annotation)
- #define rtDeclareAnnotation(variable, annotation)
- #define rtCallableProgram(return\_type, function\_name, parameter\_list) rtDeclareVariable(optix::boundCallableProgramId<return\_type parameter\_list>, function\_name,,);
- #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_MARKED_CALLABLE_PROGRAM_DEFS`
- `#define RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_NO_ARG(ReturnT)`
- `#define RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS(ReturnT,...)`
- `#define RT_INTERNAL_BOUND_CALLABLE_PROGRAM_DEFS(...)`
- `#define rtCallableProgramId optix::callableProgramId`
- `#define rtMarkedCallableProgramId optix::markedCallableProgramId`
- `#define rtCallableProgramX optix::boundCallableProgramId`

## Typedefs

- `typedef int optix::rtTextureId`

## Functions

- `__device__ int4 optix::float4AsInt4 (float4 f4)`
- `__device__ uint4 optix::float4AsUInt4 (float4 f4)`
- `template<typename ReturnT >`  
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_NO_ARG (ReturnT)`
- `template<typename ReturnT , typename Arg0T >`  
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T)`
- `template<typename ReturnT , typename Arg0T , typename Arg1T >`  
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T)`
- `template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T >`  
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T)`
- `template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T >`  
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T)`
- `template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T >`  
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T)`
- `template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T >`  
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T)`
- `template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T >`  
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T)`

- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T , typename Arg7T >  
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS` (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T)
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T , typename Arg7T , typename Arg8T >  
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS` (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T)
- template<typename ReturnT , typename Arg0T , typename Arg1T , typename Arg2T , typename Arg3T , typename Arg4T , typename Arg5T , typename Arg6T , typename Arg7T , typename Arg8T , typename Arg9T >  
`optix::RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEF_W_ARGS` (ReturnT, Arg0T, Arg1T, Arg2T, Arg3T, Arg4T, Arg5T, Arg6T, Arg7T, Arg8T, Arg9T)
- template<class T >  
  static \_\_device\_\_ void `rtTrace` (rtObject topNode, optix::Ray ray, T &prd, RTvisibilitymask mask=`RT_VISIBILITY_ALL`, RTrayflags flags=`RT_RAY_FLAG_NONE`)
- template<class T >  
  static \_\_device\_\_ void `rtTrace` (rtObject topNode, optix::Ray ray, float time, T &prd, RTvisibilitymask mask=`RT_VISIBILITY_ALL`, RTrayflags flags=`RT_RAY_FLAG_NONE`)
- 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\_\_ unsigned int `rtGetPrimitiveIndex` ()
- static \_\_device\_\_ bool `rtIsTriangleHit` ()
- static \_\_device\_\_ bool `rtIsTriangleHitBackFace` ()
- static \_\_device\_\_ bool `rtIsTriangleHitFrontFace` ()
- static \_\_device\_\_ void `rtThrow` (unsigned int code)
- static \_\_device\_\_ unsigned int `rtGetExceptionCode` ()
- static \_\_device\_\_ void `rtPrintExceptionDetails` ()
- static \_\_device\_\_ float2 `rtGetTriangleBarycentrics` ()
- 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)
- 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 , typename T13 >  
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, T13 arg13)
  
- **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< typename 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< typename 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`

### 8.15.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Host/Device side

### 8.15.2 Macro Definition Documentation

**8.15.2.1 #define \_OPTIX\_TEX\_FUNC\_DECLARE\_(  
*FUNC*,  
*SIGNATURE*,  
*PARAMS* )**

**8.15.2.2 #define RT\_CALLABLE\_PROGRAM \_\_device\_\_ \_\_noinline\_\_**

**8.15.2.3 #define RT\_INTERNAL\_BOUND\_CALLABLE\_PROGRAM\_DEFS(  
... )**

**Value:**

```
public rti_internal_callableprogram::callableProgramIdBase<__VA_ARGS__>
{
public:
 /* Default constructor */
 __device__ __forceinline__ boundCallableProgramId() {}
private:
 /* No copying of this class*/
 __device__ __forceinline__ boundCallableProgramId(const boundCallableProgramId&);
 __device__ __forceinline__ boundCallableProgramId& operator= (const boundCallableProgramId&);
}
```

**8.15.2.4 #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; }
}

```

#### 8.15.2.5 #define RT\_INTERNAL\_MARKED\_CALLABLE\_PROGRAM\_DEF\_NO\_ARG( ReturnT )

**Value:**

```

class markedCallableProgramId<ReturnT()> : public
 rti_internal_callableprogram::markedCallableProgramIdBase
 <ReturnT> \
{
 typedef callableProgramId<ReturnT()> callableProgramIdType ;
 typedef rti_internal_callableprogram::markedCallableProgramIdBase<ReturnT>
 baseType;
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEFS \
}

```

#### 8.15.2.6 #define RT\_INTERNAL\_MARKED\_CALLABLE\_PROGRAM\_DEF\_W\_ARGS( ReturnT, ... )

**Value:**

```

class markedCallableProgramId<ReturnT(__VA_ARGS__)> : public

```

```

rti_internal_callableprogram::markedCallableProgramIdBase
<ReturnT, __VA_ARGS__> \
{
 typedef callableProgramId<ReturnT(__VA_ARGS__)> callableProgramIdType; \
 typedef
 rti_internal_callableprogram::markedCallableProgramIdBase<ReturnT, __VA_ARGS__>
 BaseType; \
 RT_INTERNAL_MARKED_CALLABLE_PROGRAM_DEFS \
}

```

### 8.15.2.7 #define RT\_INTERNAL\_MARKED\_CALLABLE\_PROGRAM\_DEFS

**Value:**

```

public: \
/* Constructor that initializes the id */ \
__device__ __forceinline__ explicit markedCallableProgramId(int id, const char* callSiteName) \
: BaseType(id, callSiteName) {} \
__device__ __forceinline__ explicit markedCallableProgramId(callableProgramIdType callable, const char* \
callSiteName) \
: BaseType(callable.getId(), callSiteName) {} \
/* 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; }

```

## 8.15.3 Function Documentation

**8.15.3.1 template<class T > static \_\_device\_\_ void rtTrace (**

- rtObject *topNode*,**
- optix::Ray *ray*,**
- float *time*,**
- T & *prd*,**
- RTvisibilitymask *mask* = RT\_VISIBILITY\_ALL,**
- RTrayflags *flags* = RT\_RAY\_FLAG\_NONE ) [inline], [static]**

## 8.16 optix\_gl\_interop.h File Reference

### 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, RTgttarget target, RTtexturesampler \*textureSampler)
- RTresult RTAPI rtBufferGetGLBOId (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)

### 8.16.1 Detailed Description

OptiX public API declarations GLInterop.

Author

NVIDIA Corporation OptiX public API declarations for GL interoperability

### 8.16.2 Typedef Documentation

#### 8.16.2.1 `typedef void* HGPUNV`

## 8.17 optix\_host.h File Reference

### Macros

- #define RTAPI \_\_declspec(dllexport)

### 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 RTgeometrytriangles_api * RTgeometrytriangles`
- `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 * RTobject`
- `typedef struct RTpostprocessingstage_api * RTpostprocessingstage`
- `typedef struct RTcommandlist_api * RTcommandlist`
- `typedef int(* RTtimeoutcallback )(void)`
- `typedef void(* RTusagereportcallback )(int, const char *, const char *, void *)`

## Functions

- `RTresult RTAPI rtGetVersion (unsigned int *version)`
- `RTresult RTAPI rtGlobalSetAttribute (RTglobalattribute attrib, RTsize size, const void *p)`
- `RTresult RTAPI rtGlobalGetAttribute (RTglobalattribute attrib, RTsize size, void *p)`
- `RTresult RTAPI rtDeviceGetDeviceCount (unsigned int *count)`
- `RTresult RTAPI rtDeviceGetAttribute (int ordinal, RTdeviceattribute attrib, RTsize size, void *p)`
- `RTresult RTAPI rtVariableSetObject (RTvariable v, RTobject object)`
- `RTresult RTAPI rtVariableSetUserData (RTvariable v, RTsize size, const void *ptr)`
- `RTresult RTAPI rtVariableGetObject (RTvariable v, RTobject *object)`
- `RTresult RTAPI rtVariableGetUserData (RTvariable v, RTsize size, void *ptr)`
- `RTresult RTAPI rtVariableGetName (RTvariable v, const char **nameReturn)`
- `RTresult RTAPI rtVariableGetAnnotation (RTvariable v, const char **annotationReturn)`
- `RTresult RTAPI rtVariableGetType (RTvariable v, RTobjecttype *typeReturn)`
- `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 rtContextGetString (RTcontext context, RTresult code, const char **stringReturn)`
- `RTresult RTAPI rtContextSetAttribute (RTcontext context, RTcontextattribute attrib, RTsize size, const 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 rtContextSetStackSize (RTcontext context, RTsize bytes)`
- `RTresult RTAPI rtContextGetStackSize (RTcontext context, RTsize *bytes)`
- `RTresult RTAPI rtContextSetMaxCallableProgramDepth (RTcontext context, unsigned int maxDepth)`

- RTresult RTAPI rtContextGetMaxCallableProgramDepth (RTcontext context, unsigned int \*maxDepth)
- RTresult RTAPI rtContextSetMaxTraceDepth (RTcontext context, unsigned int maxDepth)
- RTresult RTAPI rtContextGetMaxTraceDepth (RTcontext context, unsigned int \*maxDepth)
- RTresult RTAPI rtContextSetTimeoutCallback (RTcontext context, RTtimeoutcallback callback, double minPollingSeconds)
- RTresult RTAPI rtContextSetUsageReportCallback (RTcontext context, RTusagereportcallback callback, int verbosity, void \*cbdata)
- RTresult RTAPI rtContextSetEntryPointCount (RTcontext context, unsigned int count)
- RTresult RTAPI rtContextGetEntryPointCount (RTcontext context, unsigned int \*count)
- RTresult RTAPI rtContextSetRayGenerationProgram (RTcontext context, unsigned int entryPointIndex, RTprogram program)
- RTresult RTAPI rtContextGetRayGenerationProgram (RTcontext context, unsigned int entryPointIndex, RTprogram \*program)
- RTresult RTAPI rtContextSetExceptionProgram (RTcontext context, unsigned int entryPointIndex, RTprogram program)
- RTresult RTAPI rtContextGetExceptionProgram (RTcontext context, unsigned int entryPointIndex, 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 rayTypeCount)
- RTresult RTAPI rtContextGetRayTypeCount (RTcontext context, unsigned int \*rayTypeCount)
- RTresult RTAPI rtContextSetMissProgram (RTcontext context, unsigned int rayTypeIndex, RTprogram program)
- RTresult RTAPI rtContextGetMissProgram (RTcontext context, unsigned int rayTypeIndex, RTprogram \*program)
- RTresult RTAPI rtContextGetTextureSamplerFromId (RTcontext context, int samplerId, RTtexturesampler \*sampler)
- RTresult RTAPI rtContextCompile (RTcontext context)
- RTresult RTAPI rtContextLaunch1D (RTcontext context, unsigned int entryPointIndex, RTsize width)
- RTresult RTAPI rtContextLaunch2D (RTcontext context, unsigned int entryPointIndex, RTsize width, RTsize height)
- RTresult RTAPI rtContextLaunch3D (RTcontext context, unsigned int entryPointIndex, RTsize width, RTsize height, RTsize depth)
- RTresult RTAPI rtContextGetRunningState (RTcontext context, int \*running)
- RTresult RTAPI rtContextLaunchProgressive2D (RTcontext context, unsigned int entryIndex, RTsize width, RTsize height, unsigned int maxSubframes)
- 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 bufferSizeBytes)
- RTresult RTAPI rtContextGetPrintBufferSize (RTcontext context, RTsize \*bufferSizeBytes)
- 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 \*programName, RTprogram \*program)
- RTresult RTAPI rtProgramCreateFromPTXStrings (RTcontext context, unsigned int n, const char \*\*ptxStrings, const char \*programName, RTprogram \*program)
- RTresult RTAPI rtProgramCreateFromPTXFile (RTcontext context, const char \*filename, const char \*programName, RTprogram \*program)
- RTresult RTAPI rtProgramCreateFromPTXFiles (RTcontext context, unsigned int n, const char \*\*filenames, const char \*programName, 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 \*programId)
- RTresult RTAPI rtProgramCallsiteSetPotentialCallees (RTprogram program, const char \*name, const int \*ids, int numIds)
- RTresult RTAPI rtContextGetProgramFromId (RTcontext context, int programId, 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 rtGroupSetVisibilityMask (RTgroup group, RTvisibilitymask mask)
- RTresult RTAPI rtGroupGetVisibilityMask (RTgroup group, RTvisibilitymask \*mask)
- 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 \*inverseMatrix)
- RTresult RTAPI rtTransformGetMatrix (RTtransform transform, int transpose, float \*matrix, float \*inverseMatrix)
- RTresult RTAPI rtTransformSetMotionRange (RTtransform transform, float timeBegin, float timeEnd)
- RTresult RTAPI rtTransformGetMotionRange (RTtransform transform, float \*timeBegin, float \*timeEnd)
- RTresult RTAPI rtTransformSetMotionBorderMode (RTtransform transform, RTmotionbordermode beginMode, RTmotionbordermode endMode)
- RTresult RTAPI rtTransformGetMotionBorderMode (RTtransform transform, RTmotionbordermode \*beginMode, RTmotionbordermode \*endMode)
- RTresult RTAPI rtTransformSetMotionKeys (RTtransform transform, unsigned int n, RTmotionkeytype type, const float \*keys)
- RTresult RTAPI rtTransformGetMotionKeyType (RTtransform transform, RTmotionkeytype \*type)
- RTresult RTAPI rtTransformGetMotionKeyCount (RTtransform transform, unsigned int \*n)
- RTresult RTAPI rtTransformGetMotionKeys (RTtransform transform, float \*keys)
- 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 rtGeometryGroupSetFlags (RTgeometrygroup group, RTinstanceflags flags)
- RTresult RTAPI rtGeometryGroupGetFlags (RTgeometrygroup group, RTinstanceflags \*flags)

- RTresult RTAPI rtGeometryGroupSetVisibilityMask (RTgeometrygroup group, RTvisibilitymask mask)
- RTresult RTAPI rtGeometryGroupGetVisibilityMask (RTgeometrygroup group, RTvisibilitymask \*mask)
- 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 \*\*stringReturn)
- RTresult RTAPI rtAccelerationSetTraverser (RTacceleration acceleration, const char \*traverser)
- RTresult RTAPI rtAccelerationGetTraverser (RTacceleration acceleration, const char \*\*stringReturn)
- RTresult RTAPI rtAcceleration SetProperty (RTacceleration acceleration, const char \*name, const char \*value)
- RTresult RTAPI rtAccelerationGetProperty (RTacceleration acceleration, const char \*name, const char \*\*stringReturn)
- 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 rtGeometryInstanceSetGeometryTriangles (RTgeometryinstance geometryinstance, RTgeometrytriangles geometrytriangles)
- RTresult RTAPI rtGeometryInstanceGetGeometryTriangles (RTgeometryinstance geometryinstance, RTgeometrytriangles \*geometrytriangles)
- 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 primitiveCount)
- RTresult RTAPI rtGeometryGetPrimitiveCount (RTgeometry geometry, unsigned int \*primitiveCount)
- RTresult RTAPI rtGeometrySetPrimitiveIndexOffset (RTgeometry geometry, unsigned int indexOffset)
- RTresult RTAPI rtGeometryGetPrimitiveIndexOffset (RTgeometry geometry, unsigned int \*indexOffset)
- RTresult RTAPI rtGeometrySetMotionRange (RTgeometry geometry, float timeBegin, float timeEnd)
- RTresult RTAPI rtGeometryGetMotionRange (RTgeometry geometry, float \*timeBegin, float \*timeEnd)
- RTresult RTAPI rtGeometrySetMotionBorderMode (RTgeometry geometry, RTmotionbordermode beginMode, RTmotionbordermode endMode)
- RTresult RTAPI rtGeometryGetMotionBorderMode (RTgeometry geometry, RTmotionbordermode \*beginMode, RTmotionbordermode \*endMode)
- RTresult RTAPI rtGeometrySetMotionSteps (RTgeometry geometry, unsigned int n)
- RTresult RTAPI rtGeometryGetMotionSteps (RTgeometry geometry, unsigned int \*n)
- 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 rtGeometrySetFlags (RTgeometry geometry, RTgeometryflags flags)
- RTresult RTAPI rtGeometryGetFlags (RTgeometry geometry, RTgeometryflags \*flags)

- 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 rtGeometryTrianglesCreate (RTcontext context, RTgeometrytriangles \*geometrytriangles)
- RTresult RTAPI rtGeometryTrianglesDestroy (RTgeometrytriangles geometrytriangles)
- RTresult RTAPI rtGeometryTrianglesValidate (RTgeometrytriangles geometrytriangles)
- RTresult RTAPI rtGeometryTrianglesGetContext (RTgeometrytriangles geometrytriangles, RTcontext \*context)
- RTresult RTAPI rtGeometryTrianglesSetPrimitiveIndexOffset (RTgeometrytriangles geometrytriangles, unsigned int indexOffset)
- RTresult RTAPI rtGeometryTrianglesGetPrimitiveIndexOffset (RTgeometrytriangles geometrytriangles, unsigned int \*indexOffset)
- RTresult RTAPI rtGeometryTrianglesSetPreTransformMatrix (RTgeometrytriangles geometrytriangles, int transpose, const float \*matrix)
- RTresult RTAPI rtGeometryTrianglesGetPreTransformMatrix (RTgeometrytriangles geometrytriangles, int transpose, float \*matrix)
- RTresult RTAPI rtGeometryTrianglesSetPrimitiveCount (RTgeometrytriangles geometrytriangles, unsigned int triangleCount)
- RTresult RTAPI rtGeometryTrianglesGetPrimitiveCount (RTgeometrytriangles geometrytriangles, unsigned int \*triangleCount)
- RTresult RTAPI rtGeometryTrianglesSetTriangleIndices (RTgeometrytriangles geometrytriangles, RTbuffer indexBuffer, RTsize indexBufferByteOffset, RTsize triIndicesByteStride, RTformat triIndicesFormat)
- RTresult RTAPI rtGeometryTrianglesSetVertices (RTgeometrytriangles geometrytriangles, unsigned int vertexCount, RTbuffer vertexBuffer, RTsize vertexBufferByteOffset, RTsize vertexByteStride, RTformat positionFormat)
- RTresult RTAPI rtGeometryTrianglesSetMotionVertices (RTgeometrytriangles geometrytriangles, unsigned int vertexCount, RTbuffer vertexBuffer, RTsize vertexBufferByteOffset, RTsize vertexByteStride, RTsize vertexMotionStepByteStride, RTformat positionFormat)
- RTresult RTAPI rtGeometryTrianglesSetMotionVerticesMultiBuffer (RTgeometrytriangles geometrytriangles, unsigned int vertexCount, RTbuffer \*vertexBuffers, unsigned int vertexBufferCount, RTsize vertexBufferByteOffset, RTsize vertexByteStride, RTformat positionFormat)
- RTresult RTAPI rtGeometryTrianglesSetMotionSteps (RTgeometrytriangles geometrytriangles, unsigned int motionStepCount)
- RTresult RTAPI rtGeometryTrianglesGetMotionSteps (RTgeometrytriangles geometrytriangles, unsigned int \*motionStepCount)
- RTresult RTAPI rtGeometryTrianglesSetMotionRange (RTgeometrytriangles geometrytriangles, float timeBegin, float timeEnd)

- RTresult RTAPI rtGeometryTrianglesGetMotionRange (RTgeometrytriangles geometrytriangles, float \*timeBegin, float \*timeEnd)
- RTresult RTAPI rtGeometryTrianglesSetMotionBorderMode (RTgeometrytriangles geometrytriangles, RTmotionbordermode beginMode, RTmotionbordermode endMode)
- RTresult RTAPI rtGeometryTrianglesGetMotionBorderMode (RTgeometrytriangles geometrytriangles, RTmotionbordermode \*beginMode, RTmotionbordermode \*endMode)
- RTresult RTAPI rtGeometryTrianglesSetBuildFlags (RTgeometrytriangles geometrytriangles, RTgeometrybuildflags buildFlags)
- RTresult RTAPI rtGeometryTrianglesGetMaterialCount (RTgeometrytriangles geometrytriangles, unsigned int \*numMaterials)
- RTresult RTAPI rtGeometryTrianglesSetMaterialCount (RTgeometrytriangles geometrytriangles, unsigned int numMaterials)
- RTresult RTAPI rtGeometryTrianglesSetMaterialIndices (RTgeometrytriangles geometrytriangles, RTbuffer materialIndexBuffer, RTsize materialIndexBufferByteOffset, RTsize materialIndexByteStride, RTformat materialIndexFormat)
- RTresult RTAPI rtGeometryTrianglesSetFlagsPerMaterial (RTgeometrytriangles geometrytriangles, unsigned int materialIndex, RTgeometryflags flags)
- RTresult RTAPI rtGeometryTrianglesGetFlagsPerMaterial (RTgeometrytriangles geometrytriangles, unsigned int materialIndex, RTgeometryflags \*flags)
- 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 rayTypeIndex, RTprogram program)
- RTresult RTAPI rtMaterialGetClosestHitProgram (RTmaterial material, unsigned int rayTypeIndex, RTprogram \*program)
- RTresult RTAPI rtMaterialSetAnyHitProgram (RTmaterial material, unsigned int rayTypeIndex, RTprogram program)
- RTresult RTAPI rtMaterialGetAnyHitProgram (RTmaterial material, unsigned int rayTypeIndex, 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 mipLevelCount)
- RTresult RTAPI rtTextureSamplerGetMipLevelCount (RTtexturesampler texturesampler, unsigned int \*mipLevelCount)
- RTresult RTAPI rtTextureSamplerSetArraySize (RTtexturesampler texturesampler, unsigned int textureCount)

- RTresult RTAPI rtTextureSamplerGetArraySize (RTtexturesampler texturesampler, unsigned int \*textureCount)
- 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 \*textureId)
- 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 elementSize)
- RTresult RTAPI rtBufferGetElementSize (RTbuffer buffer, RTsize \*elementSize)
- 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 \*\*userPointer)
- RTresult RTAPI rtBufferUnmap (RTbuffer buffer)
- RTresult RTAPI rtBufferMapEx (RTbuffer buffer, unsigned int mapFlags, unsigned int level, void \*userOwned, void \*\*optixOwned)
- RTresult RTAPI rtBufferUnmapEx (RTbuffer buffer, unsigned int level)
- RTresult RTAPI rtBufferGetId (RTbuffer buffer, int \*bufferId)
- RTresult RTAPI rtContextGetBufferFromId (RTcontext context, int bufferId, RTbuffer \*buffer)
- RTresult RTAPI rtBufferGetProgressiveUpdateReady (RTbuffer buffer, int \*ready, unsigned int \*subframeCount, unsigned int \*maxSubframes)
- RTresult RTAPI rtBufferBindProgressiveStream (RTbuffer stream, RTbuffer source)
- RTresult RTAPI rtBufferSetAttribute (RTbuffer buffer, RTbufferattribute attrib, RTsize size, const void \*p)
- RTresult RTAPI rtBufferGetAttribute (RTbuffer buffer, RTbufferattribute attrib, RTsize size, void \*p)
- RTresult RTAPI rtPostProcessingStageCreateBuiltin (RTcontext context, const char \*builtinName, RTpostprocessingstage \*stage)
- RTresult RTAPI rtPostProcessingStageDestroy (RTpostprocessingstage stage)
- RTresult RTAPI rtPostProcessingStageDeclareVariable (RTpostprocessingstage stage, const char \*name, RTvariable \*v)
- RTresult RTAPI rtPostProcessingStageGetContext (RTpostprocessingstage stage, RTcontext \*context)
- RTresult RTAPI rtPostProcessingStageQueryVariable (RTpostprocessingstage stage, const char \*name, RTvariable \*variable)
- RTresult RTAPI rtPostProcessingStageGetVariableCount (RTpostprocessingstage stage, unsigned int \*count)
- RTresult RTAPI rtPostProcessingStageGetVariable (RTpostprocessingstage stage, unsigned int index, RTvariable \*variable)
- RTresult RTAPI rtCommandListCreate (RTcontext context, RTcommandlist \*list)
- RTresult RTAPI rtCommandListDestroy (RTcommandlist list)
- RTresult RTAPI rtCommandListAppendPostprocessingStage (RTcommandlist list, RTpostprocessingstage stage, RTsize launchWidth, RTsize launchHeight)
- RTresult RTAPI rtCommandListAppendLaunch2D (RTcommandlist list, unsigned int entryPointIndex, RTsize launchWidth, RTsize launchHeight)
- RTresult RTAPI rtCommandListFinalize (RTcommandlist list)
- RTresult RTAPI rtCommandListExecute (RTcommandlist list)

- RTresult RTAPI rtCommandListGetContext (RTcommandlist list, RTcontext \*context)
  - RTresult RTAPI rtGeometryTrianglesSetAttributeProgram (RTgeometrytriangles geometrytriangles, RTprogram program)
  - RTresult RTAPI rtGeometryTrianglesGetAttributeProgram (RTgeometrytriangles geometrytriangles, RTprogram \*program)
  - RTresult RTAPI rtGeometryTrianglesDeclareVariable (RTgeometrytriangles geometrytriangles, const char \*name, RTvariable \*v)
  - RTresult RTAPI rtGeometryTrianglesQueryVariable (RTgeometrytriangles geometrytriangles, const char \*name, RTvariable \*v)
  - RTresult RTAPI rtGeometryTrianglesRemoveVariable (RTgeometrytriangles geometrytriangles, RTvariable v)
  - RTresult RTAPI rtGeometryTrianglesGetVariableCount (RTgeometrytriangles geometrytriangles, unsigned int \*count)
  - RTresult RTAPI rtGeometryTrianglesGetVariable (RTgeometrytriangles geometrytriangles, unsigned int index, RTvariable \*v)
- 
- 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 rtVariableSet1ll (RTvariable v, long long ll1)
  - RTresult RTAPI rtVariableSet2ll (RTvariable v, long long ll1, long long ll2)
  - RTresult RTAPI rtVariableSet3ll (RTvariable v, long long ll1, long long ll2, long long ll3)
  - RTresult RTAPI rtVariableSet4ll (RTvariable v, long long ll1, long long ll2, long long ll3, long long ll4)
  - RTresult RTAPI rtVariableSet1llv (RTvariable v, const long long \*ll)

- RTresult RTAPI rtVariableSet2llv (RTvariable v, const long long \*ll)
- RTresult RTAPI rtVariableSet3llv (RTvariable v, const long long \*ll)
- RTresult RTAPI rtVariableSet4llv (RTvariable v, const long long \*ll)
- RTresult RTAPI rtVariableSet1ull (RTvariable v, unsigned long long ull1)
- RTresult RTAPI rtVariableSet2ull (RTvariable v, unsigned long long ull1, unsigned long long ull2)
- RTresult RTAPI rtVariableSet3ull (RTvariable v, unsigned long long ull1, unsigned long long ull2, unsigned long long ull3)
- RTresult RTAPI rtVariableSet4ull (RTvariable v, unsigned long long ull1, unsigned long long ull2, unsigned long long ull3, unsigned long long ull4)
- RTresult RTAPI rtVariableSet1ullv (RTvariable v, const unsigned long long \*ull)
- RTresult RTAPI rtVariableSet2ullv (RTvariable v, const unsigned long long \*ull)
- RTresult RTAPI rtVariableSet3ullv (RTvariable v, const unsigned long long \*ull)
- RTresult RTAPI rtVariableSet4ullv (RTvariable v, const unsigned long long \*ull)
- 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 rtVariableGet1ll (RTvariable v, long long \*ll1)
- RTresult RTAPI rtVariableGet2ll (RTvariable v, long long \*ll1, long long \*ll2)
- RTresult RTAPI rtVariableGet3ll (RTvariable v, long long \*ll1, long long \*ll2, long long \*ll3)
- RTresult RTAPI rtVariableGet4ll (RTvariable v, long long \*ll1, long long \*ll2, long long \*ll3, long long \*ll4)
- RTresult RTAPI rtVariableGet1llv (RTvariable v, long long \*ll)
- RTresult RTAPI rtVariableGet2llv (RTvariable v, long long \*ll)
- RTresult RTAPI rtVariableGet3llv (RTvariable v, long long \*ll)
- RTresult RTAPI rtVariableGet4llv (RTvariable v, long long \*ll)
- RTresult RTAPI rtVariableGet1ull (RTvariable v, unsigned long long \*u1)
- RTresult RTAPI rtVariableGet2ull (RTvariable v, unsigned long long \*u1, unsigned long long \*u2)
- RTresult RTAPI rtVariableGet3ull (RTvariable v, unsigned long long \*u1, unsigned long long \*u2, unsigned long long \*u3)
- RTresult RTAPI rtVariableGet4ull (RTvariable v, unsigned long long \*u1, unsigned long long \*u2, unsigned long long \*u3, unsigned long long \*u4)
- RTresult RTAPI rtVariableGet1ullv (RTvariable v, unsigned long long \*ull)
- RTresult RTAPI rtVariableGet2ullv (RTvariable v, unsigned long long \*ull)
- RTresult RTAPI rtVariableGet3ullv (RTvariable v, unsigned long long \*ull)
- RTresult RTAPI rtVariableGet4ullv (RTvariable v, unsigned long long \*ull)
- 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)

### 8.17.1 Detailed Description

OptiX public API.

## Author

NVIDIA Corporation OptiX public API Reference - Host side

### 8.17.2 Macro Definition Documentation

#### 8.17.2.1 `#define RTAPI __declspec(dllexport)`

### 8.17.3 Typedef Documentation

#### 8.17.3.1 `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

#### 8.17.3.2 `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

#### 8.17.3.3 `typedef struct RTcommandlist_api* RTcommandlist`

Opaque type to handle CommandList - Note that the \*\_api type should never be used directly.

Only the typedef target name will be guaranteed to remain unchanged

#### 8.17.3.4 `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

#### 8.17.3.5 `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

#### 8.17.3.6 `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

#### 8.17.3.7 `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

#### 8.17.3.8 `typedef struct RTgeometrytriangles_api* RTgeometrytriangles`

Opaque type to handle GeometryTriangles - Note that the \*\_api type should never be used directly.

Only the typedef target name will be guaranteed to remain unchanged

#### **8.17.3.9 `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

#### **8.17.3.10 `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

#### **8.17.3.11 `typedef void* RTobject`**

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

#### **8.17.3.12 `typedef struct RTpostprocessingstage_api* RTpostprocessingstage`**

Opaque type to handle PostprocessingStage - Note that the \*\_api type should never be used directly.

Only the typedef target name will be guaranteed to remain unchanged

#### **8.17.3.13 `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

#### **8.17.3.14 `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

#### **8.17.3.15 `typedef unsigned int RTsize`**

#### **8.17.3.16 `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

#### **8.17.3.17 `typedef int(* RTtimeoutcallback)(void)`**

Callback signature for use with rtContextSetTimeoutCallback.

Deprecated in OptiX 6.0.

#### **8.17.3.18 `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

**8.17.3.19 `typedef void(* RTusagereportcallback)(int, const char *, const char *, void *)`**

Callback signature for use with `rtContextSetUsageReportCallback`.

**8.17.3.20 `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

## 8.17.4 Function Documentation

**8.17.4.1 `RTresult RTAPI rtAccelerationGetData (`**

`RTacceleration acceleration,`  
`void * data )`

Deprecated in OptiX 4.0.

Should not be called.

**8.17.4.2 `RTresult RTAPI rtAccelerationGetDataSize (`**

`RTacceleration acceleration,`  
`RTsize * size )`

Deprecated in OptiX 4.0.

Should not be called.

**8.17.4.3 `RTresult RTAPI rtAccelerationGetTraverser (`**

`RTacceleration acceleration,`  
`const char ** stringReturn )`

Deprecated in OptiX 4.0.

**8.17.4.4 `RTresult RTAPI rtAccelerationSetData (`**

`RTacceleration acceleration,`  
`const void * data,`  
`RTsize size )`

Deprecated in OptiX 4.0.

Should not be called.

**8.17.4.5 `RTresult RTAPI rtAccelerationSetTraverser (`**

`RTacceleration acceleration,`  
`const char * traverser )`

Deprecated in OptiX 4.0.

Setting a traverser is no longer necessary and will be ignored.

#### **8.17.4.6 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.

#### **8.17.4.7 RTResult RTAPI rtContextSetTimeoutCallback (**

**RTcontext *context*,**

**RTtimeoutcallback *callback*,**

**double *minPollingSeconds* )**

Deprecated in OptiX 6.0.

Calling this function has no effect.

#### **8.17.4.8 RTResult RTAPI rtGeometryIsDirty (**

**RTgeometry *geometry*,**

**int \* *dirty* )**

Deprecated in OptiX 4.0.

Calling this function has no effect.

#### **8.17.4.9 RTResult RTAPI rtGeometryMarkDirty (**

**RTgeometry *geometry* )**

Deprecated in OptiX 4.0.

Calling this function has no effect.

#### **8.17.4.10 RTResult RTAPI rtTextureSamplerGetArraySize (**

**RTtexturesampler *texturesampler*,**

**unsigned int \* *textureCount* )**

Deprecated in OptiX 3.9.

Use texture samplers with layered buffers instead. See [rtBufferCreate](#).

#### **8.17.4.11 RTResult RTAPI rtTextureSamplerGetMipLevelCount (**

**RTtexturesampler *texturesampler*,**

**unsigned int \* *mipLevelCount* )**

Deprecated in OptiX 3.9.

Use [rtBufferGetMipLevelCount](#) instead.

#### **8.17.4.12 RTResult RTAPI rtTextureSamplerSetArraySize (**

**RTtexturesampler *texturesampler*,**

**unsigned int *textureCount* )**

Deprecated in OptiX 3.9.

Use texture samplers with layered buffers instead. See [rtBufferCreate](#).

**8.17.4.13 RTResult RTAPI rtTextureSamplerSetMipLevelCount (**  
    **RTtexturesampler *texturesampler*,**  
    **unsigned int *mipLevelCount* )**

Deprecated in OptiX 3.9.

Use [rtBufferSetMipLevelCount](#) instead.

## 8.18 optix\_internal.h File Reference

### Namespaces

- [optix](#)

### Constant Groups

- [optix](#)

### Macros

- `#define _RT_PRINT_ACTIVE()`

### Functions

- `void optix::rt_undefined_use (int)`
- `void optix::rt_undefined_use64 (int)`
- `static __forceinline__  
 __device__ uint3 optix::rt_texture_get_size_id (int tex)`
- `static __forceinline__  
 __device__ float4 optix::rt_texture_get_gather_id (int tex, float x, float y, int comp)`
- `static __forceinline__  
 __device__ float4 optix::rt_texture_get_base_id (int tex, int dim, float x, float y, float z, int layer)`
- `static __forceinline__  
 __device__ float4 optix::rt_texture_get_level_id (int tex, int dim, float x, float y, float z, int layer,  
 float level)`
- `static __forceinline__  
 __device__ float4 optix::rt_texture_get_grad_id (int tex, int dim, float x, float y, float z, int layer,  
 float dPdx_x, float dPdx_y, float dPdx_z, float dPdy_x, float dPdy_y, float dPdy_z)`
- `static __forceinline__  
 __device__ float4 optix::rt_texture_get_f_id (int tex, int dim, float x, float y, float z, float w)`
- `static __forceinline__  
 __device__ int4 optix::rt_texture_get_i_id (int tex, int dim, float x, float y, float z, float w)`
- `static __forceinline__  
 __device__ uint4 optix::rt_texture_get_u_id (int tex, int dim, float x, float y, float z, float w)`

- static \_\_forceinline\_\_  
  \_\_device\_\_ float4 optix::rt\_texture\_get\_fetch\_id (int tex, int dim, int x, int y, int z, int w)
- static \_\_forceinline\_\_  
  \_\_device\_\_ void \* optix::rt\_buffer\_get (void \*buffer, unsigned int dim, unsigned int element\_size,  
  size\_t i0\_in, size\_t i1\_in, size\_t i2\_in, size\_t i3\_in)
- static \_\_forceinline\_\_  
  \_\_device\_\_ void \* optix::rt\_buffer\_get\_id (int id, unsigned int dim, unsigned int element\_size,  
  size\_t i0\_in, size\_t i1\_in, size\_t i2\_in, size\_t i3\_in)
- static \_\_forceinline\_\_  
  \_\_device\_\_ size\_t4 optix::rt\_buffer\_get\_size (const void \*buffer, unsigned int dim, unsigned int  
  element\_size)
- static \_\_forceinline\_\_  
  \_\_device\_\_ size\_t4 optix::rt\_buffer\_get\_size\_id (int id, unsigned int dim, unsigned int  
  element\_size)
- static \_\_forceinline\_\_  
  \_\_device\_\_ void \* optix::rt\_callable\_program\_from\_id (int id, const char \*csId=0)
- static \_\_forceinline\_\_  
  \_\_device\_\_ void optix::rt\_trace (unsigned int group, float3 origin, float3 direction, unsigned int  
  ray\_type, float tmin, float tmax, RTvisibilitymask mask, RTrayflags flags, void \*prd, unsigned int  
  prd\_size)
- static \_\_forceinline\_\_  
  \_\_device\_\_ void optix::rt\_trace\_with\_time (unsigned int group, float3 origin, float3 direction,  
  unsigned int ray\_type, float tmin, float tmax, float time, RTvisibilitymask mask, RTrayflags flags,  
  void \*prd, unsigned int prd\_size)
- static \_\_forceinline\_\_  
  \_\_device\_\_ bool optix::rt\_potential\_intersection (float t)
- static \_\_forceinline\_\_  
  \_\_device\_\_ bool optix::rt\_report\_intersection (unsigned int matlIndex)
- static \_\_forceinline\_\_  
  \_\_device\_\_ void optix::rt\_ignore\_intersection ()
- static \_\_forceinline\_\_  
  \_\_device\_\_ void optix::rt\_terminate\_ray ()
- static \_\_forceinline\_\_  
  \_\_device\_\_ void optix::rt\_intersect\_child (unsigned int index)
- static \_\_forceinline\_\_  
  \_\_device\_\_ float3 optix::rt\_transform\_point (RTtransformkind kind, const float3 &p)
- static \_\_forceinline\_\_  
  \_\_device\_\_ float3 optix::rt\_transform\_vector (RTtransformkind kind, const float3 &v)
- static \_\_forceinline\_\_  
  \_\_device\_\_ float3 optix::rt\_transform\_normal (RTtransformkind kind, const float3 &n)
- static \_\_forceinline\_\_  
  \_\_device\_\_ void optix::rt\_get\_transform (RTtransformkind kind, float matrix[16])
- static \_\_forceinline\_\_  
  \_\_device\_\_ unsigned int optix::rt\_get\_primitive\_index ()
- static \_\_forceinline\_\_  
  \_\_device\_\_ bool optix::rt\_is\_triangle\_hit ()
- static \_\_forceinline\_\_  
  \_\_device\_\_ bool optix::rt\_is\_triangle\_hit\_back\_face ()

- static \_\_forceinline\_\_  
  \_\_device\_\_ bool optix::rt\_is\_triangle\_hit\_front\_face ()
- static \_\_forceinline\_\_  
  \_\_device\_\_ void optix::rt\_throw (unsigned int code)
- static \_\_forceinline\_\_  
  \_\_device\_\_ unsigned int optix::rt\_get\_exception\_code ()
- static \_\_forceinline\_\_  
  \_\_device\_\_ int optix::rt\_print\_active ()
- static \_\_forceinline\_\_  
  \_\_device\_\_ float2 optix::rt\_get\_triangle\_barycentrics ()

### 8.18.1 Macro Definition Documentation

#### 8.18.1.1 #define \_RT\_PRINT\_ACTIVE( )

**Value:**

```
if(!optix::rt_print_active()) \
 return; \

```

## 8.19 optix\_math.h File Reference

### 8.20 optix\_prime.h File Reference

#### Macros

- #define OPTIX\_PRIME\_VERSION
- #define RTPAPI \_\_declspec(dllexport)

#### 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)

### 8.20.1 Detailed Description

OptiX Prime public API.

Author

NVIDIA Corporation OptiX Prime public API

## 8.20.2 Macro Definition Documentation

### 8.20.2.1 #define OPTIX\_PRIME\_VERSION

**Value:**

```
60000 /* major = OPTIX_PRIME_VERSION/10000, *
 * minor = (OPTIX_PRIME_VERSION%10000)/100, *
 * micro = OPTIX_PRIME_VERSION%100 */
```

### 8.20.2.2 #define RTPAPI \_\_declspec(dllimport)

## 8.20.3 Typedef Documentation

### 8.20.3.1 typedef struct CUstream\_st\* cudaStream\_t

### 8.20.3.2 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.

### 8.20.3.3 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.

### 8.20.3.4 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.

### 8.20.3.5 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.

### 8.20.3.6 typedef unsigned int RTPsize

## 8.21 optix\_prime\_declarations.h File Reference

### 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 }
```

### 8.21.1 Detailed Description

OptiX Prime public API declarations.

Author

NVIDIA Corporation OptiX Prime public API declarations

### 8.21.2 Enumeration Type Documentation

#### 8.21.2.1 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

#### 8.21.2.2 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.

### 8.21.2.3 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.

### 8.21.2.4 enum RTPcontexttype

Context types.

Enumerator

**RTP\_CONTEXT\_TYPE\_CPU** CPU context.

**RTP\_CONTEXT\_TYPE\_CUDA** CUDA context.

### 8.21.2.5 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.

### 8.21.2.6 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.

### 8.21.2.7 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.

### 8.21.2.8 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.

## 8.22 optix\_primepp.h File Reference

### Classes

- class [optix::prime::ContextObj](#)
- class [optix::prime::BufferDescObj](#)
- class [optix::prime::ModelObj](#)
- class [optix::prime::QueryObj](#)
- class [optix::prime::Exception](#)

### Namespaces

- [optix](#)
- [optix::prime](#)

### Constant Groups

- [optix](#)
- [optix::prime](#)

### 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)`

### 8.22.1 Detailed Description

A C++ wrapper around the OptiX Prime API.

### 8.22.2 Macro Definition Documentation

#### 8.22.2.1 `#define CHK(` `code ) checkError( code, getContext()->getRTPcontext() )`

## 8.23 optix\_ptx\_encryption.h File Reference

### Classes

- class `optix::PtxEncryption`

### Namespaces

- `optix`
- `optix::detail`

### Constant Groups

- `optix`
- `optix::detail`

### Macros

- `#define WIN32_LEAN_AND_MEAN 1`

## Functions

- void `optix::detail::generateSalt` (`unsigned char *buffer`)
- void `optix::detail::sha256` (`const unsigned char *input, unsigned int input_length, unsigned char *buffer`)

## Variables

- const `unsigned char optix::KEY [7] = {164, 195, 147, 255, 203, 161, 184}`

### 8.23.1 Macro Definition Documentation

#### 8.23.1.1 `#define WIN32_LEAN_AND_MEAN 1`

## 8.24 optix\_sizet.h File Reference

### Macros

- `#define RT_SIZET_INLINE static inline`
- `#define make_size_t4 make_uint4`
- `#define make_size_t3 make_uint3`
- `#define make_size_t2 make_uint2`
- `#define make_size_t1 make_uint1`

### Typedefs

- `typedef uint1 size_t1`
- `typedef uint2 size_t2`
- `typedef uint3 size_t3`
- `typedef uint4 size_t4`

### 8.24.1 Macro Definition Documentation

8.24.1.1 `#define make_size_t1 make_uint1`

8.24.1.2 `#define make_size_t2 make_uint2`

8.24.1.3 `#define make_size_t3 make_uint3`

8.24.1.4 `#define make_size_t4 make_uint4`

8.24.1.5 `#define RT_SIZET_INLINE static inline`

### 8.24.2 Typedef Documentation

8.24.2.1 `typedef uint1 size_t1`

8.24.2.2 `typedef uint2 size_t2`

8.24.2.3 `typedef uint3 size_t3`

8.24.2.4 `typedef uint4 size_t4`

## 8.25 optix\_world.h File Reference

### Macros

- `#define WIN32_LEAN_AND_MEAN`

### 8.25.1 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.

### 8.25.2 Macro Definition Documentation

8.25.2.1 `#define WIN32_LEAN_AND_MEAN`

## 8.26 optixpp.h File Reference

### Namespaces

- `optixu`

## Constant Groups

- [optixu](#)

# 8.27 optixpp\_namespace.h File Reference

## 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::GeometryTrianglesObj](#)
- class [optix::MaterialObj](#)
- class [optix::TextureSamplerObj](#)
- class [optix::BufferObj](#)
- struct [optix::bufferId< T, Dim >](#)
- class [optix::callableProgramId< T >](#)
- class [optix::markedCallableProgramId< T >](#)
- class [optix::PostprocessingStageObj](#)
- class [optix::CommandListObj](#)

## Namespaces

- [optix](#)

## Constant Groups

- [optix](#)

## Macros

- `#define WIN32_LEAN_AND_MEAN`
- `#define rtBufferId optix::bufferId`
- `#define RT_INTERNAL_CALLABLE_PROGRAM_DEFS()`
- `#define rtCallableProgramId optix::callableProgramId`
- `#define rtMarkedCallableProgramId optix::markedCallableProgramId`

## Typedefs

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

## 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 ()

### 8.27.1 Detailed Description

A C++ wrapper around the OptiX API.

### 8.27.2 Macro Definition Documentation

#### 8.27.2.1 #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;
}

```

**8.27.2.2 #define rtBufferId optix::bufferId**

**8.27.2.3 #define rtCallableProgramId optix::callableProgramId**

**8.27.2.4 #define rtMarkedCallableProgramId optix::markedCallableProgramId**

**8.27.2.5 #define WIN32\_LEAN\_AND\_MEAN**

## 8.28 optixu.h File Reference

### 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 (RTobjecttype type, char \*buffer, RTsize bufferSize)
- RTresult RTAPI rtuGetSizeForRTformat (RTformat format, size\_t \*size)
- RTresult RTAPI rtuCUDACompileString (const char \*source, const char \*\*preprocessorArguments, unsigned int numPreprocessorArguments, RTsize \*resultSize, RTsize \*errorSize)
- RTresult RTAPI rtuCUDACompileFile (const char \*filename, const char \*\*preprocessorArguments, unsigned int numPreprocessorArguments, RTsize \*resultSize, RTsize \*errorSize)
- RTresult RTAPI rtuCUDAGetCompileResult (char \*result, char \*error)
- static RTresult rtuGroupAddChild (RTgroup group, RTobject child, unsigned int \*index)
- static RTresult rtuSelectorAddChild (RTselector selector, RTobject child, unsigned int \*index)
- static RTresult rtuGeometryGroupAddChild (RTgeometrygroup geometrygroup, RTgeometryinstance child, unsigned int \*index)
- static RTresult rtuTransformSetChild (RTtransform transform, RTobject child)
- static RTresult rtuTransformGetChild (RTtransform transform, RTobject \*type)
- static RTresult rtuTransformGetChildType (RTtransform transform, RTobjecttype \*type)
- static RTresult rtuGroupRemoveChild (RTgroup group, RTobject child)
- static RTresult rtuSelectorRemoveChild (RTselector selector, RTobject child)
- static RTresult rtuGeometryGroupRemoveChild (RTgeometrygroup geometrygroup, RTgeometryinstance child)
- static RTresult rtuGroupRemoveChildByIndex (RTgroup group, unsigned int index)
- static RTresult rtuSelectorRemoveChildByIndex (RTselector selector, unsigned int index)
- static RTresult rtuGeometryGroupRemoveChildByIndex (RTgeometrygroup geometrygroup, unsigned int index)

- static RTResult rtuGroupGetChildIndex (RTgroup group, RTobject child, unsigned int \*index)
- static RTResult rtuSelectorGetChildIndex (RTselector selector, RTobject child, unsigned int \*index)
- static RTResult rtuGeometryGroupGetChildIndex (RTgeometrygroup geometrygroup, RTgeometryinstance child, unsigned int \*index)

### 8.28.1 Detailed Description

Convenience functions for the OptiX API.

### 8.28.2 Macro Definition Documentation

#### 8.28.2.1 #define RTU\_CHECK\_ERROR(     *func* )

**Value:**

```
do { \
 RTResult code = func; \
 if(code != RT_SUCCESS) \
 return code; \
} while(0)
```

#### 8.28.2.2 #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
```

#### 8.28.2.3 #define RTU\_INLINE static inline

#### 8.28.2.4 #define RTU\_SELECTOR\_ADD\_CHILD(     *\_parent*,     *\_child*,     *\_index* )

**Value:**

```
unsigned int _count; \
RTU_CHECK_ERROR(rtSelectorGetChildCount((_parent), &_count)); \
RTU_CHECK_ERROR(rtSelectorSetChildCount((_parent), _count+1)); \
RTU_CHECK_ERROR(rtSelectorSetChild((_parent), _count, (_child))); \
if(_index) *(_index) = _count; \
return RT_SUCCESS
```

## 8.29 optixu\_aabb.h File Reference

### 8.30 optixu\_aabb\_namespace.h File Reference

#### Classes

- class [optix::Aabb](#)

#### Namespaces

- [optix](#)

#### Constant Groups

- [optix](#)

#### Macros

- #define [RT\\_AABB\\_ASSERT](#) assert

#### 8.30.1 Detailed Description

OptiX public API.

##### Author

NVIDIA Corporation OptiX public API Reference - Public AABB namespace

#### 8.30.2 Macro Definition Documentation

##### 8.30.2.1 #define RT\_AABB\_ASSERT assert

### 8.31 optixu\_math.h File Reference

#### Macros

- #define [RT\\_UINT\\_USHORT\\_DEFINED](#)

## Typedefs

- `typedef unsigned int uint`
- `typedef unsigned short ushort`

### 8.31.1 Macro Definition Documentation

#### 8.31.1.1 `#define RT_UINT_USHORT_DEFINED`

### 8.31.2 Typedef Documentation

#### 8.31.2.1 `typedef unsigned int uint`

#### 8.31.2.2 `typedef unsigned short ushort`

## 8.32 optixu\_math\_namespace.h File Reference

### Classes

- `struct optix::Onb`

### Namespaces

- `optix`

### Constant Groups

- `optix`

### 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 long long optix::max (long long a, long long b)
- OPTIXU\_INLINE long long optix::min (long long a, long long b)
- OPTIXU\_INLINE unsigned int optix::max (unsigned int a, unsigned int b)
- OPTIXU\_INLINE unsigned int optix::min (unsigned int a, unsigned int b)
- OPTIXU\_INLINE unsigned long long optix::max (unsigned long long a, unsigned long long b)
- OPTIXU\_INLINE unsigned long long optix::min (unsigned long long a, unsigned long long 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**  
    long long `optix::clamp` (const long long f, const long long a, const long long b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    long long `optix::getByIndex` (const longlong1 &v, int i)
- **OPTIXU\_INLINE RT\_HOSTDEVICE void `optix::setByIndex`** (longlong1 &v, int i, long long x)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    longlong2 `optix::operator-` (const longlong2 &a)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    longlong2 `optix::min` (const longlong2 &a, const longlong2 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    longlong2 `optix::max` (const longlong2 &a, const longlong2 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    long long `optix::getByIndex` (const longlong2 &v, int i)
- **OPTIXU\_INLINE RT\_HOSTDEVICE void `optix::setByIndex`** (longlong2 &v, int i, long long x)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    longlong3 `optix::operator-` (const longlong3 &a)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    longlong3 `optix::min` (const longlong3 &a, const longlong3 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    longlong3 `optix::max` (const longlong3 &a, const longlong3 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    long long `optix::getByIndex` (const longlong3 &v, int i)
- **OPTIXU\_INLINE RT\_HOSTDEVICE void `optix::setByIndex`** (longlong3 &v, int i, int x)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    longlong4 `optix::operator-` (const longlong4 &a)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    longlong4 `optix::min` (const longlong4 &a, const longlong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    longlong4 `optix::max` (const longlong4 &a, const longlong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    long long `optix::getByIndex` (const longlong4 &v, int i)
- **OPTIXU\_INLINE RT\_HOSTDEVICE void `optix::setByIndex`** (longlong4 &v, int i, long long x)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    unsigned long long `optix::clamp` (const unsigned long long f, const unsigned long long a, const unsigned long long b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    unsigned long long `optix::getByIndex` (const ulonglong1 &v, unsigned int i)
- **OPTIXU\_INLINE RT\_HOSTDEVICE void `optix::setByIndex`** (ulonglong1 &v, int i, unsigned long long x)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    ulonglong2 `optix::min` (const ulonglong2 &a, const ulonglong2 &b)

- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong2 `optix::max` (const ulonglong2 &a, const ulonglong2 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - unsigned long long `optix::getByIndex` (const ulonglong2 &v, unsigned int i)
- **OPTIXU\_INLINE RT\_HOSTDEVICE** void `optix::setByIndex` (ulonglong2 &v, int i, unsigned long long x)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 `optix::min` (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 `optix::max` (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - unsigned long long `optix::getByIndex` (const ulonglong3 &v, unsigned int i)
- **OPTIXU\_INLINE RT\_HOSTDEVICE** void `optix::setByIndex` (ulonglong3 &v, int i, unsigned long long x)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - unsigned long long `optix::getByIndex` (const ulonglong4 &v, unsigned int i)
- **OPTIXU\_INLINE RT\_HOSTDEVICE** void `optix::setByIndex` (ulonglong4 &v, int i, unsigned long long 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 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  
longlong2 optix::make\_longlong2 (const long long s)
- OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong2 optix::make\_longlong2 (const float2 &a)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong2 optix::operator+ (const longlong2 &a, const longlong2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator+= (longlong2 &a, const longlong2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong2 optix::operator- (const longlong2 &a, const longlong2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong2 optix::operator- (const longlong2 &a, const long long b)
- OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator-= (longlong2 &a, const longlong2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong2 optix::operator\* (const longlong2 &a, const longlong2 &b)
- OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong2 optix::operator\* (const longlong2 &a, const long long s)
- OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong2 optix::operator\* (const long long s, const longlong2 &a)
- OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator\*= (longlong2 &a, const long long s)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong2 optix::clamp (const longlong2 &v, const long long a, const long long b)
- OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong2 optix::clamp (const longlong2 &v, const longlong2 &a, const longlong2 &b)
  
- OPTIXU\_INLINE RT\_HOSTDEVICE bool optix::operator== (const longlong2 &a, const longlong2 &b)

- **OPTIXU\_INLINE RT\_HOSTDEVICE** `bool optix::operator!= (const longlong2 &a, const longlong2 &b)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::make_longlong3 (const long long s)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::make_longlong3 (const float3 &a)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::operator+ (const longlong3 &a, const longlong3 &b)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE** `void optix::operator+= (longlong3 &a, const longlong3 &b)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::operator- (const longlong3 &a, const longlong3 &b)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE** `void optix::operator-= (longlong3 &a, const longlong3 &b)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::operator* (const longlong3 &a, const longlong3 &b)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::operator* (const longlong3 &a, const long long s)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::operator* (const long long s, const longlong3 &a)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE** `void optix::operator*= (longlong3 &a, const long long s)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::operator/ (const longlong3 &a, const longlong3 &b)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::operator/ (const longlong3 &a, const long long s)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::operator/ (const long long s, const longlong3 &a)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE** `void optix::operator/= (longlong3 &a, const long long s)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::clamp (const longlong3 &v, const long long a, const long long b)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::clamp (const longlong3 &v, const longlong3 &a, const longlong3 &b)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE** `bool optix::operator== (const longlong3 &a, const longlong3 &b)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE** `bool optix::operator!= (const longlong3 &a, const longlong3 &b)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong4 optix::make_longlong4 (const long long s)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong4 optix::make_longlong4 (const float4 &a)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong4 optix::operator+ (const longlong4 &a, const longlong4 &b)`

- **OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator+= (longlong4 &a, const longlong4 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong4 optix::operator- (const longlong4 &a, const longlong4 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator-= (longlong4 &a, const longlong4 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong4 optix::operator\* (const longlong4 &a, const longlong4 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong4 optix::operator\* (const longlong4 &a, const long long s)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong4 optix::operator\* (const long long s, const longlong4 &a)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator\*= (longlong4 &a, const long long s)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong4 optix::operator/ (const longlong4 &a, const longlong4 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong4 optix::operator/ (const longlong4 &a, const long long s)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong4 optix::operator/ (const long long s, const longlong4 &a)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator/= (longlong4 &a, const long long s)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong4 optix::clamp (const longlong4 &v, const long long a, const long long b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
longlong4 optix::clamp (const longlong4 &v, const longlong4 &a, const longlong4 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE bool optix::operator== (const longlong4 &a, const longlong4 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE bool optix::operator!= (const longlong4 &a, const longlong4 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
ulonglong2 optix::make\_ulonglong2 (const unsigned long long s)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
ulonglong2 optix::make\_ulonglong2 (const float2 &a)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
ulonglong2 optix::operator+ (const ulonglong2 &a, const ulonglong2 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator+= (ulonglong2 &a, const ulonglong2 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
ulonglong2 optix::operator- (const ulonglong2 &a, const ulonglong2 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
ulonglong2 optix::operator- (const ulonglong2 &a, const unsigned long long b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator-= (ulonglong2 &a, const ulonglong2 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE  
ulonglong2 optix::operator\* (const ulonglong2 &a, const ulonglong2 &b)**

- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong2 optix::operator\* (const ulonglong2 &a, const unsigned long long s)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong2 optix::operator\* (const unsigned long long s, const ulonglong2 &a)
- **OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator\*= (ulonglong2 &a, const unsigned long long s)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong2 optix::clamp (const ulonglong2 &v, const unsigned long long a, const unsigned long long b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong2 optix::clamp (const ulonglong2 &v, const ulonglong2 &a, const ulonglong2 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE bool optix::operator== (const ulonglong2 &a, const ulonglong2 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE bool optix::operator!= (const ulonglong2 &a, const ulonglong2 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 optix::make\_ulonglong3 (const unsigned long long s)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 optix::make\_ulonglong3 (const float3 &a)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 optix::operator+ (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator+= (ulonglong3 &a, const ulonglong3 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 optix::operator- (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator-= (ulonglong3 &a, const ulonglong3 &b)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 optix::operator\* (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 optix::operator\* (const ulonglong3 &a, const unsigned long long s)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 optix::operator\* (const unsigned long long s, const ulonglong3 &a)
- **OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator\*= (ulonglong3 &a, const unsigned long long s)**
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 optix::operator/ (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 optix::operator/ (const ulonglong3 &a, const unsigned long long s)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong3 optix::operator/ (const unsigned long long s, const ulonglong3 &a)
- **OPTIXU\_INLINE RT\_HOSTDEVICE void optix::operator/= (ulonglong3 &a, const unsigned long long s)**

- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong3 optix::clamp (const ulonglong3 &v, const unsigned long long a, const unsigned long long b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong3 optix::clamp (const ulonglong3 &v, const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE** bool optix::operator== (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE** bool optix::operator!= (const ulonglong3 &a, const ulonglong3 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::make\_ulonglong4 (const unsigned long long s)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::make\_ulonglong4 (const float4 &a)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::min (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::max (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::operator+ (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE** void optix::operator+= (ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::operator- (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE** void optix::operator-= (ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::operator\* (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::operator\* (const ulonglong4 &a, const unsigned long long s)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::operator\* (const unsigned long long s, const ulonglong4 &a)
- **OPTIXU\_INLINE RT\_HOSTDEVICE** void optix::operator\*= (ulonglong4 &a, const unsigned long long s)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::operator/ (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::operator/ (const ulonglong4 &a, const unsigned long long s)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
ulonglong4 optix::operator/ (const unsigned long long s, const ulonglong4 &a)
- **OPTIXU\_INLINE RT\_HOSTDEVICE** void optix::operator/= (ulonglong4 &a, const unsigned long long s)

- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong4 optix::clamp (const ulonglong4 &v, const unsigned long long a, const unsigned long long b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
  - ulonglong4 optix::clamp (const ulonglong4 &v, const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE** bool optix::operator== (const ulonglong4 &a, const ulonglong4 &b)
- **OPTIXU\_INLINE RT\_HOSTDEVICE** bool optix::operator!= (const ulonglong4 &a, const ulonglong4 &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**  
    longlong2 optix::make\_longlong2 (const longlong3 &v0)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    longlong2 optix::make\_longlong2 (const longlong4 &v0)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    longlong3 optix::make\_longlong3 (const longlong4 &v0)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    ulonglong2 optix::make\_ulonglong2 (const ulonglong3 &v0)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    ulonglong2 optix::make\_ulonglong2 (const ulonglong4 &v0)
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
    ulonglong3 optix::make\_ulonglong3 (const ulonglong4 &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**  
`longlong3 optix::make_longlong3 (const long long v0, const longlong2 &v1)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong3 optix::make_longlong3 (const longlong2 &v0, const long long v1)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong4 optix::make_longlong4 (const long long v0, const long long v1, const longlong2 &v2)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong4 optix::make_longlong4 (const long long v0, const longlong2 &v1, const long long v2)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong4 optix::make_longlong4 (const longlong2 &v0, const long long v1, const long long v2)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong4 optix::make_longlong4 (const long long v0, const longlong3 &v1)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong4 optix::make_longlong4 (const longlong3 &v0, const long long v1)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`longlong4 optix::make_longlong4 (const longlong2 &v0, const longlong2 &v1)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`ulonglong3 optix::make_ulonglong3 (const unsigned long long v0, const ulonglong2 &v1)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`ulonglong3 optix::make_ulonglong3 (const ulonglong2 &v0, const unsigned long long v1)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`ulonglong4 optix::make_ulonglong4 (const unsigned long long v0, const unsigned long long v1, const ulonglong2 &v2)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`ulonglong4 optix::make_ulonglong4 (const unsigned long long v0, const ulonglong2 &v1, const unsigned long long v2)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`ulonglong4 optix::make_ulonglong4 (const ulonglong2 &v0, const unsigned long long v1, const unsigned long long v2)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`ulonglong4 optix::make_ulonglong4 (const unsigned long long v0, const ulonglong3 &v1)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`ulonglong4 optix::make_ulonglong4 (const ulonglong3 &v0, const unsigned long long v1)`
- **OPTIXU\_INLINE RT\_HOSTDEVICE**  
`ulonglong4 optix::make_ulonglong4 (const ulonglong2 &v0, const ulonglong2 &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)`

### 8.32.1 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 modeled 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.

### 8.32.2 Macro Definition Documentation

**8.32.2.1 `#define OPTIXU_INLINE __forceinline__`**

**8.32.2.2 `#define OPTIXU_INLINE_DEFINED 1`**

**8.32.2.3 `#define OPTIXU_MATH_DEFINE_IN_NAMESPACE`**

## 8.33 optixu\_math\_stream.h File Reference

### 8.34 optixu\_math\_stream\_namespace.h File Reference

#### Namespaces

- `optix`

#### Constant Groups

- `optix`

## 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)`

### 8.34.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation Stream operators for CUDA vector types

## 8.35 optixu\_matrix.h File Reference

### 8.36 optixu\_matrix\_namespace.h File Reference

## 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 >`

## Namespaces

- `optix`

## Constant Groups

- `optix`

## Macros

- `#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 bool optix::operator==(const Matrix< M, N > &m1, const Matrix< M, N > &m2)`
- `RT_MAT_DECL OPTIXU_INLINE RT_HOSTDEVICE bool optix::operator!=(const 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, 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)`

### 8.36.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation OptiX public API Reference - Public Matrix namespace

### 8.36.2 Macro Definition Documentation

#### 8.36.2.1 `#define RT_MAT_DECL template <unsigned int M, unsigned int N>`

#### 8.36.2.2 `#define RT_MATRIX_ACCESS(`

*m*,  
*i*,  
*j* ) *m*[*i*\**N*+*j*]

## 8.37 optixu\_quaternion.h File Reference

### 8.38 optixu\_quaternion\_namespace.h File Reference

#### Classes

- class [optix::Quaternion](#)

#### Namespaces

- [optix](#)

#### Constant Groups

- [optix](#)

#### Functions

- [OPTIXU\\_INLINE RT\\_HOSTDEVICE float3 optix::operator\\* \(const Quaternion &quat, const float3 &v\)](#)
- [OPTIXU\\_INLINE RT\\_HOSTDEVICE float4 optix::operator\\* \(const Quaternion &quat, const float4 &v\)](#)
- [OPTIXU\\_INLINE RT\\_HOSTDEVICE Quaternion optix::nlerp \(const Quaternion &quat0, const Quaternion &quat1, float t\)](#)

### 8.38.1 Detailed Description

OptiX public API.

**Author**

NVIDIA Corporation OptiX public API Reference - Public QUATERNION namespace

**8.39 optixu\_vector\_functions.h File Reference****8.40 optixu\_vector\_types.h File Reference****8.41 Ref.h File Reference****Namespaces**

- optix
- optix::prime

**Constant Groups**

- optix
- optix::prime

**8.42 refman.tex File Reference**