



# NVIDIA OptiX 7.7

## API Reference Manual

12 July 2023  
Version 7.7



# Table of Contents

<b>1</b>	<b>NVIDIA OptiX 7.7 API</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>1</b>
2.1	Modules . . . . .	1
<b>3</b>	<b>Class Index</b>	<b>1</b>
3.1	Class List . . . . .	1
<b>4</b>	<b>File Index</b>	<b>4</b>
4.1	File List . . . . .	4
<b>5</b>	<b>Module Documentation</b>	<b>5</b>
5.1	Device API . . . . .	5
5.2	Function Table . . . . .	42
5.3	Host API . . . . .	43
5.4	Error handling . . . . .	44
5.5	Device context . . . . .	44
5.6	Pipelines . . . . .	44
5.7	Modules . . . . .	44
5.8	Tasks . . . . .	44
5.9	Program groups . . . . .	44
5.10	Launches . . . . .	44
5.11	Acceleration structures . . . . .	44
5.12	Denoiser . . . . .	44
5.13	Utilities . . . . .	44
5.14	Types . . . . .	50
<b>6</b>	<b>Namespace Documentation</b>	<b>100</b>
6.1	optix_impl Namespace Reference . . . . .	100
6.2	optix_internal Namespace Reference . . . . .	105
<b>7</b>	<b>Class Documentation</b>	<b>105</b>
7.1	OptixAabb Struct Reference . . . . .	105
7.2	OptixAccelBufferSizes Struct Reference . . . . .	106
7.3	OptixAccelBuildOptions Struct Reference . . . . .	106
7.4	OptixAccelEmitDesc Struct Reference . . . . .	107
7.5	OptixBuildInput Struct Reference . . . . .	108
7.6	OptixBuildInputCurveArray Struct Reference . . . . .	109
7.7	OptixBuildInputCustomPrimitiveArray Struct Reference . . . . .	111
7.8	OptixBuildInputDisplacementMicromap Struct Reference . . . . .	113
7.9	OptixBuildInputInstanceArray Struct Reference . . . . .	115
7.10	OptixBuildInputOpacityMicromap Struct Reference . . . . .	116
7.11	OptixBuildInputSphereArray Struct Reference . . . . .	117
7.12	OptixBuildInputTriangleArray Struct Reference . . . . .	119
7.13	OptixBuiltinISOOptions Struct Reference . . . . .	122
7.14	OptixDenoiserGuideLayer Struct Reference . . . . .	122
7.15	OptixDenoiserLayer Struct Reference . . . . .	123
7.16	OptixDenoiserOptions Struct Reference . . . . .	124
7.17	OptixDenoiserParams Struct Reference . . . . .	124
7.18	OptixDenoiserSizes Struct Reference . . . . .	125
7.19	OptixDeviceContextOptions Struct Reference . . . . .	126
7.20	OptixDisplacementMicromapArrayBuildInput Struct Reference . . . . .	127
7.21	OptixDisplacementMicromapDesc Struct Reference . . . . .	128

7.22	OptixDisplacementMicromapHistogramEntry Struct Reference	129
7.23	OptixDisplacementMicromapUsageCount Struct Reference	129
7.24	OptixFunctionTable Struct Reference	130
7.25	OptixImage2D Struct Reference	140
7.26	OptixInstance Struct Reference	141
7.27	OptixMatrixMotionTransform Struct Reference	142
7.28	OptixMicromapBuffers Struct Reference	143
7.29	OptixMicromapBufferSizes Struct Reference	143
7.30	OptixModuleCompileBoundValueEntry Struct Reference	144
7.31	OptixModuleCompileOptions Struct Reference	145
7.32	OptixMotionOptions Struct Reference	146
7.33	OptixOpacityMicromapArrayBuildInput Struct Reference	147
7.34	OptixOpacityMicromapDesc Struct Reference	148
7.35	OptixOpacityMicromapHistogramEntry Struct Reference	148
7.36	OptixOpacityMicromapUsageCount Struct Reference	149
7.37	OptixPayloadType Struct Reference	150
7.38	OptixPipelineCompileOptions Struct Reference	150
7.39	OptixPipelineLinkOptions Struct Reference	151
7.40	OptixProgramGroupCallables Struct Reference	152
7.41	OptixProgramGroupDesc Struct Reference	153
7.42	OptixProgramGroupHitgroup Struct Reference	154
7.43	OptixProgramGroupOptions Struct Reference	155
7.44	OptixProgramGroupSingleModule Struct Reference	155
7.45	OptixRelocateInput Struct Reference	156
7.46	OptixRelocateInputInstanceArray Struct Reference	157
7.47	OptixRelocateInputOpacityMicromap Struct Reference	157
7.48	OptixRelocateInputTriangleArray Struct Reference	157
7.49	OptixRelocationInfo Struct Reference	158
7.50	OptixShaderBindingTable Struct Reference	158
7.51	OptixSRTData Struct Reference	160
7.52	OptixSRTMotionTransform Struct Reference	162
7.53	OptixStackSizes Struct Reference	163
7.54	OptixStaticTransform Struct Reference	164
7.55	OptixUtilDenoiserImageTile Struct Reference	165
7.56	optix_internal::TypePack<... > Struct Template Reference	166
<b>8</b>	<b>File Documentation</b>	<b>166</b>
8.1	optix_device_impl.h File Reference	166
8.2	optix_device_impl.h	192
8.3	optix_device_impl_exception.h File Reference	217
8.4	optix_device_impl_exception.h	217
8.5	optix_device_impl_transformations.h File Reference	222
8.6	optix_device_impl_transformations.h	223
8.7	optix.h File Reference	230
8.8	optix.h	231
8.9	optix_denoiser_tiling.h File Reference	231
8.10	optix_denoiser_tiling.h	232
8.11	optix_device.h File Reference	237
8.12	optix_device.h	242
8.13	optix_function_table.h File Reference	249
8.14	optix_function_table.h	250
8.15	optix_function_table_definition.h File Reference	255
8.16	optix_function_table_definition.h	255

8.17	optix_host.h File Reference . . . . .	256
8.18	optix_host.h . . . . .	282
8.19	optix_stack_size.h File Reference . . . . .	287
8.20	optix_stack_size.h . . . . .	288
8.21	optix_stubs.h File Reference . . . . .	292
8.22	optix_stubs.h . . . . .	293
8.23	optix_types.h File Reference . . . . .	304
8.24	optix_types.h . . . . .	314
8.25	main.dox File Reference . . . . .	334

# 1 NVIDIA OptiX 7.7 API

This document describes the NVIDIA OptiX 7.7 application programming interface. See <https://raytracing-docs.nvidia.com/> for more information about programming with NVIDIA OptiX.

## 2 Module Index

### 2.1 Modules

Here is a list of all modules:

Device API	5
Function Table	42
Host API	43
Error handling	44
Device context	44
Pipelines	44
Modules	44
Tasks	44
Program groups	44
Launches	44
Acceleration structures	44
Denoiser	44
Utilities	44
Types	50

## 3 Class Index

### 3.1 Class List

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

<code>OptixAabb</code>	
AABB inputs	105
<code>OptixAccelBufferSizes</code>	
Struct for querying builder allocation requirements	106
<code>OptixAccelBuildOptions</code>	
Build options for acceleration structures	106
<code>OptixAccelEmitDesc</code>	
Specifies a type and output destination for emitted post-build properties	107
<code>OptixBuildInput</code>	
Build inputs	108
<code>OptixBuildInputCurveArray</code>	
Curve inputs	109
<code>OptixBuildInputCustomPrimitiveArray</code>	
Custom primitive inputs	111

<code>OptixBuildInputDisplacementMicromap</code>	Optional displacement part of a triangle array input	113
<code>OptixBuildInputInstanceArray</code>	Instance and instance pointer inputs	115
<code>OptixBuildInputOpacityMicromap</code>		116
<code>OptixBuildInputSphereArray</code>	Sphere inputs	117
<code>OptixBuildInputTriangleArray</code>	Triangle inputs	119
<code>OptixBuiltinISOOptions</code>	Specifies the options for retrieving an intersection program for a built-in primitive type. The primitive type must not be <code>OPTIX_PRIMITIVE_TYPE_CUSTOM</code>	122
<code>OptixDenoiserGuideLayer</code>	Guide layer for the denoiser	122
<code>OptixDenoiserLayer</code>	Input/Output layers for the denoiser	123
<code>OptixDenoiserOptions</code>	Options used by the denoiser	124
<code>OptixDenoiserParams</code>		124
<code>OptixDenoiserSizes</code>	Various sizes related to the denoiser	125
<code>OptixDeviceContextOptions</code>	Parameters used for <code>optixDeviceContextCreate()</code>	126
<code>OptixDisplacementMicromapArrayBuildInput</code>	Inputs to displacement micromaps array construction	127
<code>OptixDisplacementMicromapDesc</code>		128
<code>OptixDisplacementMicromapHistogramEntry</code>	Displacement micromap histogram entry. Specifies how many displacement micromaps of a specific type are input to the displacement micromap array build. Note that while this is similar to <code>OptixDisplacementMicromapUsageCount</code> , the histogram entry specifies how many displacement micromaps of a specific type are combined into a displacement micromap array	129
<code>OptixDisplacementMicromapUsageCount</code>	Displacement micromap usage count for acceleration structure builds. Specifies how many displacement micromaps of a specific type are referenced by triangles when building the AS. Note that while this is similar to <code>OptixDisplacementMicromapHistogramEntry</code> , the usage count specifies how many displacement micromaps of a specific type are referenced by triangles in the AS	129
<code>OptixFunctionTable</code>	The function table containing all API functions	130
<code>OptixImage2D</code>	Image descriptor used by the denoiser	140
<code>OptixInstance</code>	Instances	141

<code>OptixMatrixMotionTransform</code>	Represents a matrix motion transformation	142
<code>OptixMicromapBuffers</code>	Buffer inputs for opacity/displacement micromap array builds	143
<code>OptixMicromapBufferSizes</code>	Conservative memory requirements for building a opacity/displacement micromap array	143
<code>OptixModuleCompileBoundValueEntry</code>	Struct for specifying specializations for pipelineParams as specified in <code>OptixPipelineCompileOptions::pipelineLaunchParamsVariableName</code>	144
<code>OptixModuleCompileOptions</code>	Compilation options for module	145
<code>OptixMotionOptions</code>	Motion options	146
<code>OptixOpacityMicromapArrayBuildInput</code>	Inputs to opacity micromap array construction	147
<code>OptixOpacityMicromapDesc</code>	Opacity micromap descriptor	148
<code>OptixOpacityMicromapHistogramEntry</code>	Opacity micromap histogram entry. Specifies how many opacity micromaps of a specific type are input to the opacity micromap array build. Note that while this is similar to <code>OptixOpacityMicromapUsageCount</code> , the histogram entry specifies how many opacity micromaps of a specific type are combined into a opacity micromap array	148
<code>OptixOpacityMicromapUsageCount</code>	Opacity micromap usage count for acceleration structure builds. Specifies how many opacity micromaps of a specific type are referenced by triangles when building the AS. Note that while this is similar to <code>OptixOpacityMicromapHistogramEntry</code> , the usage count specifies how many opacity micromaps of a specific type are referenced by triangles in the AS	149
<code>OptixPayloadType</code>	Specifies a single payload type	150
<code>OptixPipelineCompileOptions</code>	Compilation options for all modules of a pipeline	150
<code>OptixPipelineLinkOptions</code>	Link options for a pipeline	151
<code>OptixProgramGroupCallables</code>	Program group representing callables	152
<code>OptixProgramGroupDesc</code>	Descriptor for program groups	153
<code>OptixProgramGroupHitgroup</code>	Program group representing the hitgroup	154
<code>OptixProgramGroupOptions</code>	Program group options	155

<code>OptixProgramGroupSingleModule</code>	Program group representing a single module	155
<code>OptixRelocateInput</code>	Relocation inputs	156
<code>OptixRelocateInputInstanceArray</code>	Instance and instance pointer inputs	157
<code>OptixRelocateInputOpacityMicromap</code>		157
<code>OptixRelocateInputTriangleArray</code>	Triangle inputs	157
<code>OptixRelocationInfo</code>	Used to store information related to relocation of optix data structures	158
<code>OptixShaderBindingTable</code>	Describes the shader binding table (SBT)	158
<code>OptixSRTData</code>	Represents an SRT transformation	160
<code>OptixSRTMotionTransform</code>	Represents an SRT motion transformation	162
<code>OptixStackSizes</code>	Describes the stack size requirements of a program group	163
<code>OptixStaticTransform</code>	Static transform	164
<code>OptixUtilDenoiserImageTile</code>	Tile definition	165
<code>optix_internal::TypePack&lt;... &gt;</code>		166

## 4 File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<code>optix_device_impl.h</code>	OptiX public API	166
<code>optix_device_impl_exception.h</code>	OptiX public API	217
<code>optix_device_impl_transformations.h</code>	OptiX public API	222
<code>optix.h</code>	OptiX public API header	230
<code>optix_denoiser_tiling.h</code>	OptiX public API header	231
<code>optix_device.h</code>	OptiX public API header	237



<a href="#">optix_function_table.h</a>	
OptiX public API header	249
<a href="#">optix_function_table_definition.h</a>	
OptiX public API header	255
<a href="#">optix_host.h</a>	
OptiX public API header	256
<a href="#">optix_stack_size.h</a>	
OptiX public API header	287
<a href="#">optix_stubs.h</a>	
OptiX public API header	292
<a href="#">optix_types.h</a>	
OptiX public API header	304

## 5 Module Documentation

### 5.1 Device API

#### Functions

- `template<typename... Payload>`  
`static __forceinline__ __device__ void optixTrace (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBTOffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)`
- `template<typename... Payload>`  
`static __forceinline__ __device__ void optixTrace (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBTOffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)`
- `static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_5 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_6 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_7 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_8 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_9 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_10 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_11 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_12 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_13 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_14 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_15 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_16 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_17 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_18 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_19 (unsigned int p)`

- NVIDIA OptiX 7.7 API

- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetRayTmin ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetRayTmax ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetRayTime ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetRayFlags ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetRayVisibilityMask ()
- static \_\_forceinline\_\_ \_\_device\_\_ OptixTraversableHandle optixGetInstanceTraversableFromIAS (OptixTraversableHandle ias, unsigned int instIdx)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetTriangleVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetMicroTriangleVertexData (float3 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetMicroTriangleBarycentricsData (float2 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetLinearCurveVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetQuadraticBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetCubicBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetCatmullRomVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetCubicBezierVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetRibbonVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixGetRibbonNormal (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetSphereData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static \_\_forceinline\_\_ \_\_device\_\_ OptixTraversableHandle optixGetGASTraversableHandle ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetGASMotionTimeBegin (OptixTraversableHandle gas)
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetGASMotionTimeEnd (OptixTraversableHandle gas)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetGASMotionStepCount (OptixTraversableHandle gas)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetWorldToObjectTransformMatrix (float m[12])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetObjectToWorldTransformMatrix (float m[12])
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformPointFromWorldToObjectSpace (float3 point)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformVectorFromWorldToObjectSpace (float3 vec)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformNormalFromWorldToObjectSpace (float3 normal)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformPointFromObjectToWorldSpace (float3 point)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformVectorFromObjectToWorldSpace (float3 vec)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformNormalFromObjectToWorldSpace (float3 normal)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetTransformListSize ()

- static `__forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle` (unsigned int index)
- static `__forceinline__ __device__ OptixTransformType optixGetTransformTypeFromHandle` (OptixTraversableHandle handle)
- static `__forceinline__ __device__ const OptixStaticTransform *`  
`optixGetStaticTransformFromHandle` (OptixTraversableHandle handle)
- static `__forceinline__ __device__ const OptixSRTMotionTransform *`  
`optixGetSRTMotionTransformFromHandle` (OptixTraversableHandle handle)
- static `__forceinline__ __device__ const OptixMatrixMotionTransform *`  
`optixGetMatrixMotionTransformFromHandle` (OptixTraversableHandle handle)
- static `__forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle` (OptixTraversableHandle handle)
- static `__forceinline__ __device__ OptixTraversableHandle optixGetInstanceChildFromHandle` (OptixTraversableHandle handle)
- static `__forceinline__ __device__ const float4 * optixGetInstanceTransformFromHandle` (OptixTraversableHandle handle)
- static `__forceinline__ __device__ const float4 * optixGetInstanceInverseTransformFromHandle` (OptixTraversableHandle handle)
- static `__forceinline__ __device__ bool optixReportIntersection` (float hitT, unsigned int hitKind)
- static `__forceinline__ __device__ bool optixReportIntersection` (float hitT, unsigned int hitKind, unsigned int a0)
- static `__forceinline__ __device__ bool optixReportIntersection` (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1)
- static `__forceinline__ __device__ bool optixReportIntersection` (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2)
- static `__forceinline__ __device__ bool optixReportIntersection` (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3)
- static `__forceinline__ __device__ bool optixReportIntersection` (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
- static `__forceinline__ __device__ bool optixReportIntersection` (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5)
- static `__forceinline__ __device__ bool optixReportIntersection` (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6)
- static `__forceinline__ __device__ bool optixReportIntersection` (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6, unsigned int a7)
- static `__forceinline__ __device__ unsigned int optixGetAttribute_0` ()
- static `__forceinline__ __device__ unsigned int optixGetAttribute_1` ()
- static `__forceinline__ __device__ unsigned int optixGetAttribute_2` ()
- static `__forceinline__ __device__ unsigned int optixGetAttribute_3` ()
- static `__forceinline__ __device__ unsigned int optixGetAttribute_4` ()
- static `__forceinline__ __device__ unsigned int optixGetAttribute_5` ()
- static `__forceinline__ __device__ unsigned int optixGetAttribute_6` ()
- static `__forceinline__ __device__ unsigned int optixGetAttribute_7` ()
- static `__forceinline__ __device__ void optixTerminateRay` ()
- static `__forceinline__ __device__ void optixIgnoreIntersection` ()
- static `__forceinline__ __device__ unsigned int optixGetPrimitiveIndex` ()
- static `__forceinline__ __device__ unsigned int optixGetSbtGASIndex` ()
- static `__forceinline__ __device__ unsigned int optixGetInstanceId` ()

- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetInstanceIndex ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetHitKind ()
- static \_\_forceinline\_\_ \_\_device\_\_ OptixPrimitiveType optixGetPrimitiveType (unsigned int hitKind)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsFrontFaceHit (unsigned int hitKind)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsBackFaceHit (unsigned int hitKind)
- static \_\_forceinline\_\_ \_\_device\_\_ OptixPrimitiveType optixGetPrimitiveType ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsFrontFaceHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsBackFaceHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsTriangleHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsTriangleFrontFaceHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsTriangleBackFaceHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsDisplacedMicromeshTriangleHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsDisplacedMicromeshTriangleFrontFaceHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsDisplacedMicromeshTriangleBackFaceHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ float2 optixGetTriangleBarycentrics ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetCurveParameter ()
- static \_\_forceinline\_\_ \_\_device\_\_ float2 optixGetRibbonParameters ()
- static \_\_forceinline\_\_ \_\_device\_\_ uint3 optixGetLaunchIndex ()
- static \_\_forceinline\_\_ \_\_device\_\_ uint3 optixGetLaunchDimensions ()
- static \_\_forceinline\_\_ \_\_device\_\_ CUdeviceptr optixGetSbtDataPointer ()
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6, unsigned int exceptionDetail7)
- static \_\_forceinline\_\_ \_\_device\_\_ int optixGetExceptionCode ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_0 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_1 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_2 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_3 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_4 ()



- `static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5 ()`
- `static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6 ()`
- `static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7 ()`
- `static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable ()`
- `static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset ()`
- `static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay ()`
- `static __forceinline__ __device__ OptixParameterMismatchExceptionDetails optixGetExceptionParameterMismatch ()`
- `static __forceinline__ __device__ char * optixGetExceptionLineInfo ()`
- `template<typename ReturnT , typename... ArgTypes>`  
`static __forceinline__ __device__ ReturnT optixDirectCall (unsigned int sbtIndex, ArgTypes... args)`
- `template<typename ReturnT , typename... ArgTypes>`  
`static __forceinline__ __device__ ReturnT optixContinuationCall (unsigned int sbtIndex, ArgTypes... args)`
- `static __forceinline__ __device__ uint4 optixTexFootprint2D (unsigned long long tex, unsigned int texInfo, float x, float y, unsigned int *singleMipLevel)`
- `static __forceinline__ __device__ uint4 optixTexFootprint2DLod (unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool coarse, unsigned int *singleMipLevel)`
- `static __forceinline__ __device__ uint4 optixTexFootprint2DGrad (unsigned long long tex, unsigned int texInfo, float x, float y, float dPdx_x, float dPdx_y, float dPdy_x, float dPdy_y, bool coarse, unsigned int *singleMipLevel)`

## 5.1.1 Detailed Description

OptiX Device API.

## 5.1.2 Function Documentation

### 5.1.2.1 optixContinuationCall()

```
template<typename ReturnT , typename... ArgTypes>
static __forceinline__ __device__ ReturnT optixContinuationCall (
    unsigned int sbtIndex,
    ArgTypes... args ) [static]
```

Creates a call to the continuation callable program at the specified SBT entry.

This will call the program that was specified in the `OptixProgramGroupCallables::entryFunctionNameCC` in the module specified by `OptixProgramGroupCallables::moduleCC`. The address of the SBT entry is calculated by `OptixShaderBindingTable::callablesRecordBase + (OptixShaderBindingTable::callablesRecordStrideInBytes * sbtIndex)`. As opposed to direct callable programs, continuation callable programs are allowed to call `optixTrace` recursively.

Behavior is undefined if there is no continuation callable program at the specified SBT entry.

Behavior is undefined if the number of arguments that are being passed in does not match the number of parameters expected by the program that is called. In that case an exception of type `OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH` will be thrown if `OPTIX_EXCEPTION_FLAG_DEBUG` was specified for the `OptixPipelineCompileOptions::exceptionFlags`.

## Parameters

in	<i>sbtIndex</i>	The offset of the SBT entry of the continuation callable program to call relative to <a href="#">OptixShaderBindingTable::callablesRecordBase</a> .
in	<i>args</i>	The arguments to pass to the continuation callable program.

## 5.1.2.2 optixDirectCall()

```
template<typename ReturnT , typename... ArgTypes>
static __forceinline__ __device__ ReturnT optixDirectCall (
    unsigned int sbtIndex,
    ArgTypes... args ) [static]
```

Creates a call to the direct callable program at the specified SBT entry.

This will call the program that was specified in the [OptixProgramGroupCallables::entryFunctionNameDC](#) in the module specified by [OptixProgramGroupCallables::moduleDC](#). The address of the SBT entry is calculated by [OptixShaderBindingTable::callablesRecordBase](#) + ([OptixShaderBindingTable::callablesRecordStrideInBytes](#) \* *sbtIndex*).

Behavior is undefined if there is no direct callable program at the specified SBT entry.

Behavior is undefined if the number of arguments that are being passed in does not match the number of parameters expected by the program that is called. In that case an exception of type `OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH` will be thrown if `OPTIX_EXCEPTION_FLAG_DEBUG` was specified for the [OptixPipelineCompileOptions::exceptionFlags](#).

## Parameters

in	<i>sbtIndex</i>	The offset of the SBT entry of the direct callable program to call relative to <a href="#">OptixShaderBindingTable::callablesRecordBase</a> .
in	<i>args</i>	The arguments to pass to the direct callable program.

## 5.1.2.3 optixGetAttribute\_0()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_0 ( ) [static]
```

Returns the attribute at slot 0.

## 5.1.2.4 optixGetAttribute\_1()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_1 ( ) [static]
```

Returns the attribute at slot 1.

## 5.1.2.5 optixGetAttribute\_2()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_2 ( ) [static]
```

Returns the attribute at slot 2.

## 5.1.2.6 optixGetAttribute\_3()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_3 ( ) [static]
```

Returns the attribute at slot 3.

### 5.1.2.7 optixGetAttribute\_4()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_4 ( ) [static]
```

Returns the attribute at slot 4.

### 5.1.2.8 optixGetAttribute\_5()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_5 ( ) [static]
```

Returns the attribute at slot 5.

### 5.1.2.9 optixGetAttribute\_6()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_6 ( ) [static]
```

Returns the attribute at slot 6.

### 5.1.2.10 optixGetAttribute\_7()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_7 ( ) [static]
```

Returns the attribute at slot 7.

### 5.1.2.11 optixGetCatmullRomVertexData()

```
static __forceinline__ __device__ void optixGetCatmullRomVertexData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
    float4 data[4] ) [static]
```

Return the object space curve control vertex data of a CatmullRom spline curve in a Geometry Acceleration Structure (GAS) at a given motion time. To access vertex data, the GAS must be built using the flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS`.

$data[i] = \{x,y,z,w\}$  with  $\{x,y,z\}$  the position and  $w$  the radius of control vertex  $i$ . If motion is disabled via [OptixPipelineCompileOptions::usesMotionBlur](#), or the GAS does not contain motion, the time parameter is ignored.

### 5.1.2.12 optixGetCubicBezierVertexData()

```
static __forceinline__ __device__ void optixGetCubicBezierVertexData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
    float4 data[4] ) [static]
```

Return the object space curve control vertex data of a cubic Bezier curve in a Geometry Acceleration Structure (GAS) at a given motion time. To access vertex data, the GAS must be built using the flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS`.

$data[i] = \{x,y,z,w\}$  with  $\{x,y,z\}$  the position and  $w$  the radius of control vertex  $i$ . If motion is disabled via [OptixPipelineCompileOptions::usesMotionBlur](#), or the GAS does not contain motion, the time parameter is ignored.



### 5.1.2.13 optixGetCubicBSplineVertexData()

```
static __forceinline__ __device__ void optixGetCubicBSplineVertexData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
    float4 data[4] ) [static]
```

Return the object space curve control vertex data of a cubic BSpline curve in a Geometry Acceleration Structure (GAS) at a given motion time. To access vertex data, the GAS must be built using the flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS`.

`data[i] = {x,y,z,w}` with `{x,y,z}` the position and `w` the radius of control vertex `i`. If motion is disabled via `OptixPipelineCompileOptions::usesMotionBlur`, or the GAS does not contain motion, the time parameter is ignored.

### 5.1.2.14 optixGetCurveParameter()

```
static __forceinline__ __device__ float optixGetCurveParameter ( ) [static]
```

Returns the curve parameter associated with the current intersection when using `OptixBuildInputCurveArray` objects.

### 5.1.2.15 optixGetExceptionCode()

```
static __forceinline__ __device__ int optixGetExceptionCode ( ) [static]
```

Returns the exception code.

Only available in EX.

### 5.1.2.16 optixGetExceptionDetail\_0()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0 ( )
[static]
```

Returns the 32-bit exception detail at slot 0.

The behavior is undefined if the exception is not a user exception, or the used overload `optixThrowException()` did not provide the queried exception detail.

Only available in EX.

### 5.1.2.17 optixGetExceptionDetail\_1()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1 ( )
[static]
```

Returns the 32-bit exception detail at slot 1.

See also `optixGetExceptionDetail_0()`

### 5.1.2.18 optixGetExceptionDetail\_2()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2 ( )
[static]
```

Returns the 32-bit exception detail at slot 2.

See also [optixGetExceptionDetail\\_0\(\)](#)

#### 5.1.2.19 optixGetExceptionDetail\_3()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3 ( )  
[static]
```

Returns the 32-bit exception detail at slot 3.

See also [optixGetExceptionDetail\\_0\(\)](#)

#### 5.1.2.20 optixGetExceptionDetail\_4()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4 ( )  
[static]
```

Returns the 32-bit exception detail at slot 4.

See also [optixGetExceptionDetail\\_0\(\)](#)

#### 5.1.2.21 optixGetExceptionDetail\_5()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5 ( )  
[static]
```

Returns the 32-bit exception detail at slot 5.

See also [optixGetExceptionDetail\\_0\(\)](#)

#### 5.1.2.22 optixGetExceptionDetail\_6()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6 ( )  
[static]
```

Returns the 32-bit exception detail at slot 6.

See also [optixGetExceptionDetail\\_0\(\)](#)

#### 5.1.2.23 optixGetExceptionDetail\_7()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7 ( )  
[static]
```

Returns the 32-bit exception detail at slot 7.

See also [optixGetExceptionDetail\\_0\(\)](#)

#### 5.1.2.24 optixGetExceptionInvalidRay()

```
static __forceinline__ __device__ OptixInvalidRayExceptionDetails  
optixGetExceptionInvalidRay ( ) [static]
```

Returns the invalid ray for exceptions with exception code `OPTIX_EXCEPTION_CODE_INVALID_RAY`. Exceptions of type `OPTIX_EXCEPTION_CODE_INVALID_RAY` are thrown when one or more values that were passed into `optixTrace` are either `inf` or `nan`.

`OptixInvalidRayExceptionDetails::rayTime` will always be 0 if [OptixPipelineCompileOptions::usesMotionBlur](#) is 0. Values in the returned struct are all zero for all other exception codes.

Only available in EX.

### 5.1.2.25 optixGetExceptionInvalidSbtOffset()

```
static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset ( )  
[static]
```

Returns the invalid sbt offset for exceptions with exception code `OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT` and `OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT`.

Returns zero for all other exception codes.

Only available in EX.

### 5.1.2.26 optixGetExceptionInvalidTraversable()

```
static __forceinline__ __device__ OptixTraversableHandle  
optixGetExceptionInvalidTraversable ( ) [static]
```

Returns the invalid traversable handle for exceptions with exception code `OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE`.

Returns zero for all other exception codes.

Only available in EX.

### 5.1.2.27 optixGetExceptionLineInfo()

```
static __forceinline__ __device__ char * optixGetExceptionLineInfo ( ) [static]
```

Returns a string that includes information about the source location that caused the current exception.

The source location is only available for exceptions of type `OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH`, `OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE`, `OPTIX_EXCEPTION_CODE_INVALID_RAY`, and for user exceptions. Line information needs to be present in the input PTX and `OptixModuleCompileOptions::debugLevel` may not be set to `OPTIX_COMPILE_DEBUG_LEVEL_NONE`.

Returns a NULL pointer if no line information is available.

Only available in EX.

### 5.1.2.28 optixGetExceptionParameterMismatch()

```
static __forceinline__ __device__ OptixParameterMismatchExceptionDetails  
optixGetExceptionParameterMismatch ( ) [static]
```

Returns information about an exception with code `OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH`.

Exceptions of type `OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH` are called when the number of arguments that were passed into a call to `optixDirectCall` or `optixContinuationCall` does not match the number of parameters of the callable that is called. Note that the parameters are packed by OptiX into individual 32 bit values, so the number of expected and passed values may not correspond to the number of arguments passed into `optixDirectCall` or `optixContinuationCall`.

Values in the returned struct are all zero for all other exception codes.

Only available in EX.

### 5.1.2.29 optixGetGASMotionStepCount()

```
static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (
```

`OptixTraversableHandle gas ) [static]`

Returns the number of motion steps of a GAS (see [OptixMotionOptions](#))

#### 5.1.2.30 optixGetGASMotionTimeBegin()

`static __forceinline__ __device__ float optixGetGASMotionTimeBegin (   
 OptixTraversableHandle gas ) [static]`

Returns the motion begin time of a GAS (see [OptixMotionOptions](#))

#### 5.1.2.31 optixGetGASMotionTimeEnd()

`static __forceinline__ __device__ float optixGetGASMotionTimeEnd (   
 OptixTraversableHandle gas ) [static]`

Returns the motion end time of a GAS (see [OptixMotionOptions](#))

#### 5.1.2.32 optixGetGASTraversableHandle()

`static __forceinline__ __device__ OptixTraversableHandle   
 optixGetGASTraversableHandle ( ) [static]`

Returns the traversable handle for the Geometry Acceleration Structure (GAS) containing the current hit. May be called from IS, AH and CH.

#### 5.1.2.33 optixGetHitKind()

`static __forceinline__ __device__ unsigned int optixGetHitKind ( ) [static]`

Returns the 8 bit hit kind associated with the current hit.

Use [optixGetPrimitiveType\(\)](#) to interpret the hit kind. For custom intersections (primitive type `OPTIX_PRIMITIVE_TYPE_CUSTOM`), this is the 7-bit hitKind passed to [optixReportIntersection\(\)](#). Hit kinds greater than 127 are reserved for built-in primitives.

Available only in AH and CH.

#### 5.1.2.34 optixGetInstanceChildFromHandle()

`static __forceinline__ __device__ OptixTraversableHandle   
 optixGetInstanceChildFromHandle (   
 OptixTraversableHandle handle ) [static]`

Returns child traversable handle from an [OptixInstance](#) traversable.

Returns 0 if the traversable handle does not reference an [OptixInstance](#).

#### 5.1.2.35 optixGetInstanceId()

`static __forceinline__ __device__ unsigned int optixGetInstanceId ( ) [static]`

Returns the [OptixInstance::instanceId](#) of the instance within the top level acceleration structure associated with the current intersection.

When building an acceleration structure using [OptixBuildInputInstanceArray](#) each [OptixInstance](#) has a user supplied instanceId. [OptixInstance](#) objects reference another acceleration structure. During traversal the acceleration structures are visited top down. In the IS and AH programs the [OptixInstance::instanceId](#) corresponding to the most recently visited [OptixInstance](#) is returned when calling [optixGetInstanceId\(\)](#). In CH [optixGetInstanceId\(\)](#) returns the [OptixInstance::instanceId](#) when

the hit was recorded with `optixReportIntersection`. In the case where there is no `OptixInstance` visited, `optixGetInstanceId` returns `~0u`

#### 5.1.2.36 `optixGetInstanceIdFromHandle()`

```
static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle
(
    OptixTraversableHandle handle ) [static]
```

Returns `instanceId` from an `OptixInstance` traversable.

Returns 0 if the traversable handle does not reference an `OptixInstance`.

#### 5.1.2.37 `optixGetInstanceIndex()`

```
static __forceinline__ __device__ unsigned int optixGetInstanceIndex ( )
[static]
```

Returns the zero-based index of the instance within its instance acceleration structure associated with the current intersection.

In the IS and AH programs the index corresponding to the most recently visited `OptixInstance` is returned when calling `optixGetInstanceIndex()`. In CH `optixGetInstanceIndex()` returns the index when the hit was recorded with `optixReportIntersection`. In the case where there is no `OptixInstance` visited, `optixGetInstanceIndex` returns 0

#### 5.1.2.38 `optixGetInstanceInverseTransformFromHandle()`

```
static __forceinline__ __device__ const float4 *
optixGetInstanceInverseTransformFromHandle (
    OptixTraversableHandle handle ) [static]
```

Returns world-to-object transform from an `OptixInstance` traversable.

Returns 0 if the traversable handle does not reference an `OptixInstance`.

#### 5.1.2.39 `optixGetInstanceTransformFromHandle()`

```
static __forceinline__ __device__ const float4 *
optixGetInstanceTransformFromHandle (
    OptixTraversableHandle handle ) [static]
```

Returns object-to-world transform from an `OptixInstance` traversable.

Returns 0 if the traversable handle does not reference an `OptixInstance`.

#### 5.1.2.40 `optixGetInstanceTraversableFromIAS()`

```
static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS (
    OptixTraversableHandle ias,
    unsigned int instIdx ) [static]
```

Return the traversable handle of a given instance in an Instance Acceleration Structure (IAS) To obtain instance traversables by index, the IAS must be built using the flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS`.

#### 5.1.2.41 optixGetLaunchDimensions()

```
static __forceinline__ __device__ uint3 optixGetLaunchDimensions ( ) [static]
```

Available in any program, it returns the dimensions of the current launch specified by `optixLaunch` on the host.

#### 5.1.2.42 optixGetLaunchIndex()

```
static __forceinline__ __device__ uint3 optixGetLaunchIndex ( ) [static]
```

Available in any program, it returns the current launch index within the launch dimensions specified by `optixLaunch` on the host.

The raygen program is typically only launched once per launch index.

#### 5.1.2.43 optixGetLinearCurveVertexData()

```
static __forceinline__ __device__ void optixGetLinearCurveVertexData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
    float4 data[2] ) [static]
```

Return the object space curve control vertex data of a linear curve in a Geometry Acceleration Structure (GAS) at a given motion time. To access vertex data, the GAS must be built using the flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS`.

`data[i] = {x,y,z,w}` with `{x,y,z}` the position and `w` the radius of control vertex `i`. If motion is disabled via `OptixPipelineCompileOptions::usesMotionBlur`, or the GAS does not contain motion, the time parameter is ignored.

#### 5.1.2.44 optixGetMatrixMotionTransformFromHandle()

```
static __forceinline__ __device__ const OptixMatrixMotionTransform *
optixGetMatrixMotionTransformFromHandle (
    OptixTraversableHandle handle ) [static]
```

Returns a pointer to a `OptixMatrixMotionTransform` from its traversable handle.

Returns 0 if the traversable is not of type `OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM`.

#### 5.1.2.45 optixGetMicroTriangleBarycentricsData()

```
static __forceinline__ __device__ void optixGetMicroTriangleBarycentricsData
(
    float2 data[3] ) [static]
```

Returns the barycentrics of the vertices of the currently intersected micro triangle with respect to the base triangle.

#### 5.1.2.46 optixGetMicroTriangleVertexData()

```
static __forceinline__ __device__ void optixGetMicroTriangleVertexData (
    float3 data[3] ) [static]
```

Return the object space micro triangle vertex positions of the current hit. The current hit must be a displacement micromap triangle hit.

#### 5.1.2.47 optixGetObjectRayDirection()

```
static __forceinline__ __device__ float3 optixGetObjectRayDirection ( )  
[static]
```

Returns the current object space ray direction based on the current transform stack.

Only available in IS and AH.

#### 5.1.2.48 optixGetObjectRayOrigin()

```
static __forceinline__ __device__ float3 optixGetObjectRayOrigin ( ) [static]
```

Returns the current object space ray origin based on the current transform stack.

Only available in IS and AH.

#### 5.1.2.49 optixGetObjectToWorldTransformMatrix()

```
static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix  
(  
    float m[12] ) [static]
```

Returns the object-to-world transformation matrix resulting from the current active transformation list.

The cost of this function may be proportional to the size of the transformation list.

#### 5.1.2.50 optixGetPayload\_0()

```
static __forceinline__ __device__ unsigned int optixGetPayload_0 ( ) [static]
```

Reads the 32-bit payload value at slot 0.

#### 5.1.2.51 optixGetPayload\_1()

```
static __forceinline__ __device__ unsigned int optixGetPayload_1 ( ) [static]
```

Reads the 32-bit payload value at slot 1.

#### 5.1.2.52 optixGetPayload\_10()

```
static __forceinline__ __device__ unsigned int optixGetPayload_10 ( ) [static]
```

Reads the 32-bit payload value at slot 10.

#### 5.1.2.53 optixGetPayload\_11()

```
static __forceinline__ __device__ unsigned int optixGetPayload_11 ( ) [static]
```

Reads the 32-bit payload value at slot 11.

#### 5.1.2.54 optixGetPayload\_12()

```
static __forceinline__ __device__ unsigned int optixGetPayload_12 ( ) [static]
```

Reads the 32-bit payload value at slot 12.

#### 5.1.2.55 optixGetPayload\_13()

```
static __forceinline__ __device__ unsigned int optixGetPayload_13 ( ) [static]
```

Reads the 32-bit payload value at slot 13.

#### 5.1.2.56 optixGetPayload\_14()

```
static __forceinline__ __device__ unsigned int optixGetPayload_14 ( ) [static]
```

Reads the 32-bit payload value at slot 14.

#### 5.1.2.57 optixGetPayload\_15()

```
static __forceinline__ __device__ unsigned int optixGetPayload_15 ( ) [static]
```

Reads the 32-bit payload value at slot 15.

#### 5.1.2.58 optixGetPayload\_16()

```
static __forceinline__ __device__ unsigned int optixGetPayload_16 ( ) [static]
```

Reads the 32-bit payload value at slot 16.

#### 5.1.2.59 optixGetPayload\_17()

```
static __forceinline__ __device__ unsigned int optixGetPayload_17 ( ) [static]
```

Reads the 32-bit payload value at slot 17.

#### 5.1.2.60 optixGetPayload\_18()

```
static __forceinline__ __device__ unsigned int optixGetPayload_18 ( ) [static]
```

Reads the 32-bit payload value at slot 18.

#### 5.1.2.61 optixGetPayload\_19()

```
static __forceinline__ __device__ unsigned int optixGetPayload_19 ( ) [static]
```

Reads the 32-bit payload value at slot 19.

#### 5.1.2.62 optixGetPayload\_2()

```
static __forceinline__ __device__ unsigned int optixGetPayload_2 ( ) [static]
```

Reads the 32-bit payload value at slot 2.

#### 5.1.2.63 optixGetPayload\_20()

```
static __forceinline__ __device__ unsigned int optixGetPayload_20 ( ) [static]
```

Reads the 32-bit payload value at slot 20.

#### 5.1.2.64 optixGetPayload\_21()

```
static __forceinline__ __device__ unsigned int optixGetPayload_21 ( ) [static]
```

Reads the 32-bit payload value at slot 21.



#### 5.1.2.65 optixGetPayload\_22()

```
static __forceinline__ __device__ unsigned int optixGetPayload_22 ( ) [static]
```

Reads the 32-bit payload value at slot 22.

#### 5.1.2.66 optixGetPayload\_23()

```
static __forceinline__ __device__ unsigned int optixGetPayload_23 ( ) [static]
```

Reads the 32-bit payload value at slot 23.

#### 5.1.2.67 optixGetPayload\_24()

```
static __forceinline__ __device__ unsigned int optixGetPayload_24 ( ) [static]
```

Reads the 32-bit payload value at slot 24.

#### 5.1.2.68 optixGetPayload\_25()

```
static __forceinline__ __device__ unsigned int optixGetPayload_25 ( ) [static]
```

Reads the 32-bit payload value at slot 25.

#### 5.1.2.69 optixGetPayload\_26()

```
static __forceinline__ __device__ unsigned int optixGetPayload_26 ( ) [static]
```

Reads the 32-bit payload value at slot 26.

#### 5.1.2.70 optixGetPayload\_27()

```
static __forceinline__ __device__ unsigned int optixGetPayload_27 ( ) [static]
```

Reads the 32-bit payload value at slot 27.

#### 5.1.2.71 optixGetPayload\_28()

```
static __forceinline__ __device__ unsigned int optixGetPayload_28 ( ) [static]
```

Reads the 32-bit payload value at slot 28.

#### 5.1.2.72 optixGetPayload\_29()

```
static __forceinline__ __device__ unsigned int optixGetPayload_29 ( ) [static]
```

Reads the 32-bit payload value at slot 29.

#### 5.1.2.73 optixGetPayload\_3()

```
static __forceinline__ __device__ unsigned int optixGetPayload_3 ( ) [static]
```

Reads the 32-bit payload value at slot 3.

#### 5.1.2.74 optixGetPayload\_30()

```
static __forceinline__ __device__ unsigned int optixGetPayload_30 ( ) [static]
```

Reads the 32-bit payload value at slot 30.

### 5.1.2.75 optixGetPayload\_31()

```
static __forceinline__ __device__ unsigned int optixGetPayload_31 ( ) [static]
```

Reads the 32-bit payload value at slot 31.

### 5.1.2.76 optixGetPayload\_4()

```
static __forceinline__ __device__ unsigned int optixGetPayload_4 ( ) [static]
```

Reads the 32-bit payload value at slot 4.

### 5.1.2.77 optixGetPayload\_5()

```
static __forceinline__ __device__ unsigned int optixGetPayload_5 ( ) [static]
```

Reads the 32-bit payload value at slot 5.

### 5.1.2.78 optixGetPayload\_6()

```
static __forceinline__ __device__ unsigned int optixGetPayload_6 ( ) [static]
```

Reads the 32-bit payload value at slot 6.

### 5.1.2.79 optixGetPayload\_7()

```
static __forceinline__ __device__ unsigned int optixGetPayload_7 ( ) [static]
```

Reads the 32-bit payload value at slot 7.

### 5.1.2.80 optixGetPayload\_8()

```
static __forceinline__ __device__ unsigned int optixGetPayload_8 ( ) [static]
```

Reads the 32-bit payload value at slot 8.

### 5.1.2.81 optixGetPayload\_9()

```
static __forceinline__ __device__ unsigned int optixGetPayload_9 ( ) [static]
```

Reads the 32-bit payload value at slot 9.

### 5.1.2.82 optixGetPrimitiveIndex()

```
static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ( )  
[static]
```

For a given [OptixBuildInputTriangleArray](#) the number of primitives is defined as "(  
OptixBuildInputTriangleArray::indexBuffer == 0) ? OptixBuildInputTriangleArray::numVertices/3 :  
OptixBuildInputTriangleArray::numIndexTriplets;". For a given  
[OptixBuildInputCustomPrimitiveArray](#) the number of primitives is defined as numAabbs.

The primitive index returns the index into the array of primitives plus the primitiveIndexOffset.

In IS and AH this corresponds to the currently intersected primitive. In CH this corresponds to the primitive index of the closest intersected primitive.

### 5.1.2.83 optixGetPrimitiveType() [1/2]

```
static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ( )  
[static]
```

Function interpreting the hit kind associated with the current `optixReportIntersection`.

#### 5.1.2.84 `optixGetPrimitiveType()` [2/2]

```
static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (
    unsigned int hitKind ) [static]
```

Function interpreting the result of `optixGetHitKind()`.

#### 5.1.2.85 `optixGetQuadraticBSplineVertexData()`

```
static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
    float4 data[3] ) [static]
```

Return the object space curve control vertex data of a quadratic BSpline curve in a Geometry Acceleration Structure (GAS) at a given motion time. To access vertex data, the GAS must be built using the flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS`.

`data[i] = {x,y,z,w}` with `{x,y,z}` the position and `w` the radius of control vertex `i`. If motion is disabled via `OptixPipelineCompileOptions::usesMotionBlur`, or the GAS does not contain motion, the time parameter is ignored.

#### 5.1.2.86 `optixGetRayFlags()`

```
static __forceinline__ __device__ unsigned int optixGetRayFlags ( ) [static]
```

Returns the `rayFlags` passed into `optixTrace`.

Only available in IS, AH, CH, MS

#### 5.1.2.87 `optixGetRayTime()`

```
static __forceinline__ __device__ float optixGetRayTime ( ) [static]
```

Returns the `rayTime` passed into `optixTrace`.

Will return 0 if motion is disabled. Only available in IS, AH, CH, MS

#### 5.1.2.88 `optixGetRayTmax()`

```
static __forceinline__ __device__ float optixGetRayTmax ( ) [static]
```

In IS and CH returns the current smallest reported `hitT` or the `tmax` passed into `optixTrace` if no hit has been reported In AH returns the `hitT` value as passed in to `optixReportIntersection` In MS returns the `tmax` passed into `optixTrace` Only available in IS, AH, CH, MS.

#### 5.1.2.89 `optixGetRayTmin()`

```
static __forceinline__ __device__ float optixGetRayTmin ( ) [static]
```

Returns the `tmin` passed into `optixTrace`.

Only available in IS, AH, CH, MS

### 5.1.2.90 optixGetRayVisibilityMask()

```
static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask ( )  
[static]
```

Returns the visibilityMask passed into optixTrace.

Only available in IS, AH, CH, MS

### 5.1.2.91 optixGetRibbonNormal()

```
static __forceinline__ __device__ float3 optixGetRibbonNormal (  
    OptixTraversableHandle gas,  
    unsigned int primIdx,  
    unsigned int sbtGASIndex,  
    float time,  
    float2 ribbonParameters ) [static]
```

Return ribbon normal at intersection reported by optixReportIntersection.

### 5.1.2.92 optixGetRibbonParameters()

```
static __forceinline__ __device__ float2 optixGetRibbonParameters ( ) [static]
```

Returns the ribbon parameters along directrix (length) and generator (width) of the current intersection when using [OptixBuildInputCurveArray](#) objects with curveType OPTIX\_PRIMITIVE\_TYPE\_FLAT\_QUADRATIC\_BSPLINE.

### 5.1.2.93 optixGetRibbonVertexData()

```
static __forceinline__ __device__ void optixGetRibbonVertexData (  
    OptixTraversableHandle gas,  
    unsigned int primIdx,  
    unsigned int sbtGASIndex,  
    float time,  
    float4 data[3] ) [static]
```

Return the object space curve control vertex data of a ribbon (flat quadratic BSpline) in a Geometry Acceleration Structure (GAS) at a given motion time. To access vertex data, the GAS must be built using the flag OPTIX\_BUILD\_FLAG\_ALLOW\_RANDOM\_VERTEX\_ACCESS.

data[i] = {x,y,z,w} with {x,y,z} the position and w the radius of control vertex i. If motion is disabled via [OptixPipelineCompileOptions::usesMotionBlur](#), or the GAS does not contain motion, the time parameter is ignored.

### 5.1.2.94 optixGetSbtDataPointer()

```
static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ( )  
[static]
```

Returns the generic memory space pointer to the data region (past the header) of the currently active SBT record corresponding to the current program.

### 5.1.2.95 optixGetSbtGASIndex()

```
static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ( ) [static]
```

Returns the Sbt GAS index of the primitive associated with the current intersection.

In IS and AH this corresponds to the currently intersected primitive. In CH this corresponds to the Sbt GAS index of the closest intersected primitive. In EX with exception code `OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT` corresponds to the sbt index within the hit GAS. Returns zero for all other exceptions.

#### 5.1.2.96 optixGetSphereData()

```
static __forceinline__ __device__ void optixGetSphereData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
    float4 data[1] ) [static]
```

Return the object space sphere data, center point and radius, in a Geometry Acceleration Structure (GAS) at a given motion time. To access sphere data, the GAS must be built using the flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS`.

`data[0] = {x,y,z,w}` with `{x,y,z}` the position of the sphere center and `w` the radius. If motion is disabled via `OptixPipelineCompileOptions::usesMotionBlur`, or the GAS does not contain motion, the time parameter is ignored.

#### 5.1.2.97 optixGetSRTMotionTransformFromHandle()

```
static __forceinline__ __device__ const OptixSRTMotionTransform *
optixGetSRTMotionTransformFromHandle (
    OptixTraversableHandle handle ) [static]
```

Returns a pointer to a `OptixSRTMotionTransform` from its traversable handle.

Returns 0 if the traversable is not of type `OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM`.

#### 5.1.2.98 optixGetStaticTransformFromHandle()

```
static __forceinline__ __device__ const OptixStaticTransform *
optixGetStaticTransformFromHandle (
    OptixTraversableHandle handle ) [static]
```

Returns a pointer to a `OptixStaticTransform` from its traversable handle.

Returns 0 if the traversable is not of type `OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM`.

#### 5.1.2.99 optixGetTransformListHandle()

```
static __forceinline__ __device__ OptixTraversableHandle
optixGetTransformListHandle (
    unsigned int index ) [static]
```

Returns the traversable handle for a transform on the current transform list.

Only available in IS, AH, CH, EX

### 5.1.2.100 optixGetTransformListSize()

```
static __forceinline__ __device__ unsigned int optixGetTransformListSize ( )  
[static]
```

Returns the number of transforms on the current transform list.

Only available in IS, AH, CH, EX

### 5.1.2.101 optixGetTransformTypeFromHandle()

```
static __forceinline__ __device__ OptixTransformType  
optixGetTransformTypeFromHandle (   
    OptixTraversableHandle handle ) [static]
```

Returns the transform type of a traversable handle from a transform list.

### 5.1.2.102 optixGetTriangleBarycentrics()

```
static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ( )  
[static]
```

Convenience function that returns the first two attributes as floats.

When using [OptixBuildInputTriangleArray](#) objects, during intersection the barycentric coordinates are stored into the first two attribute registers.

### 5.1.2.103 optixGetTriangleVertexData()

```
static __forceinline__ __device__ void optixGetTriangleVertexData (   
    OptixTraversableHandle gas,  
    unsigned int primIdx,  
    unsigned int sbtGASIndex,  
    float time,  
    float3 data[3] ) [static]
```

Return the object space triangle vertex positions of a given triangle in a Geometry Acceleration Structure (GAS) at a given motion time. To access vertex data, the GAS must be built using the flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS`.

If motion is disabled via [OptixPipelineCompileOptions::usesMotionBlur](#), or the GAS does not contain motion, the time parameter is ignored.

### 5.1.2.104 optixGetWorldRayDirection()

```
static __forceinline__ __device__ float3 optixGetWorldRayDirection ( ) [static]
```

Returns the rayDirection passed into optixTrace.

May be more expensive to call in IS and AH than their object space counterparts, so effort should be made to use the object space ray in those programs. Only available in IS, AH, CH, MS

### 5.1.2.105 optixGetWorldRayOrigin()

```
static __forceinline__ __device__ float3 optixGetWorldRayOrigin ( ) [static]
```

Returns the rayOrigin passed into optixTrace.

May be more expensive to call in IS and AH than their object space counterparts, so effort should be

made to use the object space ray in those programs. Only available in IS, AH, CH, MS

#### 5.1.2.106 optixGetWorldToObjectTransformMatrix()

```
static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix
(
    float m[12] ) [static]
```

Returns the world-to-object transformation matrix resulting from the current active transformation list. The cost of this function may be proportional to the size of the transformation list.

#### 5.1.2.107 optixIgnoreIntersection()

```
static __forceinline__ __device__ void optixIgnoreIntersection ( ) [static]
```

Discards the hit, and returns control to the calling `optixReportIntersection` or built-in intersection routine.

Available only in AH.

#### 5.1.2.108 optixIsBackFaceHit() [1/2]

```
static __forceinline__ __device__ bool optixIsBackFaceHit ( ) [static]
```

Function interpreting the hit kind associated with the current `optixReportIntersection`.

#### 5.1.2.109 optixIsBackFaceHit() [2/2]

```
static __forceinline__ __device__ bool optixIsBackFaceHit (
    unsigned int hitKind ) [static]
```

Function interpreting the result of `optixGetHitKind()`.

#### 5.1.2.110 optixIsDisplacedMicromeshTriangleBackFaceHit()

```
static __forceinline__ __device__ bool
optixIsDisplacedMicromeshTriangleBackFaceHit ( ) [static]
```

Convenience function interpreting the result of `optixGetHitKind()`.

#### 5.1.2.111 optixIsDisplacedMicromeshTriangleFrontFaceHit()

```
static __forceinline__ __device__ bool
optixIsDisplacedMicromeshTriangleFrontFaceHit ( ) [static]
```

Convenience function interpreting the result of `optixGetHitKind()`.

#### 5.1.2.112 optixIsDisplacedMicromeshTriangleHit()

```
static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit
( ) [static]
```

Convenience function interpreting the result of `optixGetHitKind()`.

#### 5.1.2.113 optixIsFrontFaceHit() [1/2]

```
static __forceinline__ __device__ bool optixIsFrontFaceHit ( ) [static]
```

Function interpreting the hit kind associated with the current `optixReportIntersection`.

5.1.2.114 `optixIsFrontFaceHit()` [2/2]

```
static __forceinline__ __device__ bool optixIsFrontFaceHit (
    unsigned int hitKind ) [static]
```

Function interpreting the result of `optixGetHitKind()`.

5.1.2.115 `optixIsTriangleBackFaceHit()`

```
static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ( ) [static]
```

Convenience function interpreting the result of `optixGetHitKind()`.

5.1.2.116 `optixIsTriangleFrontFaceHit()`

```
static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ( ) [static]
```

Convenience function interpreting the result of `optixGetHitKind()`.

5.1.2.117 `optixIsTriangleHit()`

```
static __forceinline__ __device__ bool optixIsTriangleHit ( ) [static]
```

Convenience function interpreting the result of `optixGetHitKind()`.

5.1.2.118 `optixReportIntersection()` [1/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind ) [static]
```

Reports an intersections (overload without attributes).

If `optixGetRayTmin() <= hitT <= optixGetRayTmax()`, the any hit program associated with this intersection program (via the SBT entry) is called. The AH program can do one of three things:

1. call `optixIgnoreIntersection` - no hit is recorded, `optixReportIntersection` returns false
2. call `optixTerminateRay` - hit is recorded, `optixReportIntersection` does not return, no further traversal occurs, and the associated closest hit program is called
3. neither - hit is recorded, `optixReportIntersection` returns true `hitKind` - Only the 7 least significant bits should be written [0..127]. Any values above 127 are reserved for built in intersection. The value can be queried with `optixGetHitKind()` in AH and CH.

The attributes specified with `a0..a7` are available in the AH and CH programs. Note that the attributes available in the CH program correspond to the closest recorded intersection. The number of attributes in registers and memory can be configured in the pipeline.

## Parameters

in	<i>hitT</i>
in	<i>hitKind</i>

5.1.2.119 `optixReportIntersection()` [2/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
```



```
float hitT,
unsigned int hitKind,
unsigned int a0 ) [static]
```

Reports an intersection (overload with 1 attribute register).

See also [optixReportIntersection\(float,unsigned int\)](#)

#### 5.1.2.120 optixReportIntersection() [3/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1 ) [static]
```

Reports an intersection (overload with 2 attribute registers).

See also [optixReportIntersection\(float,unsigned int\)](#)

#### 5.1.2.121 optixReportIntersection() [4/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1,
    unsigned int a2 ) [static]
```

Reports an intersection (overload with 3 attribute registers).

See also [optixReportIntersection\(float,unsigned int\)](#)

#### 5.1.2.122 optixReportIntersection() [5/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1,
    unsigned int a2,
    unsigned int a3 ) [static]
```

Reports an intersection (overload with 4 attribute registers).

See also [optixReportIntersection\(float,unsigned int\)](#)

#### 5.1.2.123 optixReportIntersection() [6/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
```

```

    unsigned int a1,
    unsigned int a2,
    unsigned int a3,
    unsigned int a4 ) [static]

```

Reports an intersection (overload with 5 attribute registers).

See also [optixReportIntersection\(float,unsigned int\)](#)

#### 5.1.2.124 optixReportIntersection() [7/9]

```

static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1,
    unsigned int a2,
    unsigned int a3,
    unsigned int a4,
    unsigned int a5 ) [static]

```

Reports an intersection (overload with 6 attribute registers).

See also [optixReportIntersection\(float,unsigned int\)](#)

#### 5.1.2.125 optixReportIntersection() [8/9]

```

static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1,
    unsigned int a2,
    unsigned int a3,
    unsigned int a4,
    unsigned int a5,
    unsigned int a6 ) [static]

```

Reports an intersection (overload with 7 attribute registers).

See also [optixReportIntersection\(float,unsigned int\)](#)

#### 5.1.2.126 optixReportIntersection() [9/9]

```

static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1,
    unsigned int a2,

```

```

        unsigned int a3,
        unsigned int a4,
        unsigned int a5,
        unsigned int a6,
        unsigned int a7 ) [static]

```

Reports an intersection (overload with 8 attribute registers).

See also [optixReportIntersection\(float,unsigned int\)](#)

#### 5.1.2.127 optixSetPayload\_0()

```

static __forceinline__ __device__ void optixSetPayload_0 (
    unsigned int p ) [static]

```

Writes the 32-bit payload value at slot 0.

#### 5.1.2.128 optixSetPayload\_1()

```

static __forceinline__ __device__ void optixSetPayload_1 (
    unsigned int p ) [static]

```

Writes the 32-bit payload value at slot 1.

#### 5.1.2.129 optixSetPayload\_10()

```

static __forceinline__ __device__ void optixSetPayload_10 (
    unsigned int p ) [static]

```

Writes the 32-bit payload value at slot 10.

#### 5.1.2.130 optixSetPayload\_11()

```

static __forceinline__ __device__ void optixSetPayload_11 (
    unsigned int p ) [static]

```

Writes the 32-bit payload value at slot 11.

#### 5.1.2.131 optixSetPayload\_12()

```

static __forceinline__ __device__ void optixSetPayload_12 (
    unsigned int p ) [static]

```

Writes the 32-bit payload value at slot 12.

#### 5.1.2.132 optixSetPayload\_13()

```

static __forceinline__ __device__ void optixSetPayload_13 (
    unsigned int p ) [static]

```

Writes the 32-bit payload value at slot 13.

#### 5.1.2.133 optixSetPayload\_14()

```

static __forceinline__ __device__ void optixSetPayload_14 (
    unsigned int p ) [static]

```

Writes the 32-bit payload value at slot 14.

#### 5.1.2.134 optixSetPayload\_15()

```
static __forceinline__ __device__ void optixSetPayload_15 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 15.

#### 5.1.2.135 optixSetPayload\_16()

```
static __forceinline__ __device__ void optixSetPayload_16 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 16.

#### 5.1.2.136 optixSetPayload\_17()

```
static __forceinline__ __device__ void optixSetPayload_17 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 17.

#### 5.1.2.137 optixSetPayload\_18()

```
static __forceinline__ __device__ void optixSetPayload_18 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 18.

#### 5.1.2.138 optixSetPayload\_19()

```
static __forceinline__ __device__ void optixSetPayload_19 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 19.

#### 5.1.2.139 optixSetPayload\_2()

```
static __forceinline__ __device__ void optixSetPayload_2 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 2.

#### 5.1.2.140 optixSetPayload\_20()

```
static __forceinline__ __device__ void optixSetPayload_20 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 20.

#### 5.1.2.141 optixSetPayload\_21()

```
static __forceinline__ __device__ void optixSetPayload_21 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 21.

#### 5.1.2.142 optixSetPayload\_22()

```
static __forceinline__ __device__ void optixSetPayload_22 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 22.

#### 5.1.2.143 optixSetPayload\_23()

```
static __forceinline__ __device__ void optixSetPayload_23 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 23.

#### 5.1.2.144 optixSetPayload\_24()

```
static __forceinline__ __device__ void optixSetPayload_24 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 24.

#### 5.1.2.145 optixSetPayload\_25()

```
static __forceinline__ __device__ void optixSetPayload_25 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 25.

#### 5.1.2.146 optixSetPayload\_26()

```
static __forceinline__ __device__ void optixSetPayload_26 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 26.

#### 5.1.2.147 optixSetPayload\_27()

```
static __forceinline__ __device__ void optixSetPayload_27 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 27.

#### 5.1.2.148 optixSetPayload\_28()

```
static __forceinline__ __device__ void optixSetPayload_28 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 28.

#### 5.1.2.149 optixSetPayload\_29()

```
static __forceinline__ __device__ void optixSetPayload_29 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 29.

#### 5.1.2.150 optixSetPayload\_3()

```
static __forceinline__ __device__ void optixSetPayload_3 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 3.

#### 5.1.2.151 optixSetPayload\_30()

```
static __forceinline__ __device__ void optixSetPayload_30 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 30.

#### 5.1.2.152 optixSetPayload\_31()

```
static __forceinline__ __device__ void optixSetPayload_31 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 31.

#### 5.1.2.153 optixSetPayload\_4()

```
static __forceinline__ __device__ void optixSetPayload_4 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 4.

#### 5.1.2.154 optixSetPayload\_5()

```
static __forceinline__ __device__ void optixSetPayload_5 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 5.

#### 5.1.2.155 optixSetPayload\_6()

```
static __forceinline__ __device__ void optixSetPayload_6 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 6.

#### 5.1.2.156 optixSetPayload\_7()

```
static __forceinline__ __device__ void optixSetPayload_7 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 7.

#### 5.1.2.157 optixSetPayload\_8()

```
static __forceinline__ __device__ void optixSetPayload_8 (  
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 8.

5.1.2.158 `optixSetPayload_9()`

```
static __forceinline__ __device__ void optixSetPayload_9 (
    unsigned int p ) [static]
```

Writes the 32-bit payload value at slot 9.

5.1.2.159 `optixSetPayloadTypes()`

```
static __forceinline__ __device__ void optixSetPayloadTypes (
    unsigned int typeMask ) [static]
```

Specify the supported payload types for a program.

The supported types are specified as a bitwise combination of payload types. (See `OptixPayloadTypeID`) May only be called once per program. Must be called at the top of the program. Only available in IS, AH, CH, MS

5.1.2.160 `optixTerminateRay()`

```
static __forceinline__ __device__ void optixTerminateRay ( ) [static]
```

Record the hit, stops traversal, and proceeds to CH.

Available only in AH.

5.1.2.161 `optixTexFootprint2D()`

```
static __forceinline__ __device__ uint4 optixTexFootprint2D (
    unsigned long long tex,
    unsigned int texInfo,
    float x,
    float y,
    unsigned int * singleMipLevel ) [static]
```

`optixTexFootprint2D` calculates the footprint of a corresponding 2D texture fetch (non-mipmapped).

On Turing and subsequent architectures, a texture footprint instruction allows user programs to determine the set of texels that would be accessed by an equivalent filtered texture lookup.

## Parameters

in	<i>tex</i>	CUDA texture object (cast to 64-bit integer)
in	<i>texInfo</i>	Texture info packed into 32-bit integer, described below.
in	<i>x</i>	Texture coordinate
in	<i>y</i>	Texture coordinate
out	<i>singleMipLevel</i>	Result indicating whether the footprint spans only a single miplevel.

The texture info argument is a packed 32-bit integer with the following layout:

`texInfo[31:29]` = reserved (3 bits) `texInfo[28:24]` = miplevel count (5 bits) `texInfo[23:20]` = log2 of tile width (4 bits) `texInfo[19:16]` = log2 of tile height (4 bits) `texInfo[15:10]` = reserved (6 bits) `texInfo[9:8]` = horizontal wrap mode (2 bits) (`CUaddress_mode`) `texInfo[7:6]` = vertical wrap mode (2 bits) (`CUaddress_mode`) `texInfo[5]` = mipmap filter mode (1 bit) (`CUfilter_mode`) `texInfo[4:0]` = maximum anisotropy (5 bits)

Returns a 16-byte structure (as a uint4) that stores the footprint of a texture request at a particular "granularity", which has the following layout:

```
struct Texture2DFootprint { unsigned long long mask; unsigned int tileY : 12; unsigned int reserved1 : 4; unsigned int dx : 3; unsigned int dy : 3; unsigned int reserved2 : 2; unsigned int granularity : 4; unsigned int reserved3 : 4; unsigned int tileX : 12; unsigned int level : 4; unsigned int reserved4 : 16; };
```

The granularity indicates the size of texel groups that are represented by an 8x8 bitmask. For example, a granularity of 12 indicates texel groups that are 128x64 texels in size. In a footprint call, The returned granularity will either be the actual granularity of the result, or 0 if the footprint call was able to honor the requested granularity (the usual case).

level is the mip level of the returned footprint. Two footprint calls are needed to get the complete footprint when a texture call spans multiple mip levels.

mask is an 8x8 bitmask of texel groups that are covered, or partially covered, by the footprint. tileX and tileY give the starting position of the mask in 8x8 texel-group blocks. For example, suppose a granularity of 12 (128x64 texels), and tileX=3 and tileY=4. In this case, bit 0 of the mask (the low order bit) corresponds to texel group coordinates (3\*8, 4\*8), and texel coordinates (3\*8\*128, 4\*8\*64), within the specified mip level.

If nonzero, dx and dy specify a "toroidal rotation" of the bitmask. Toroidal rotation of a coordinate in the mask simply means that its value is reduced by 8. Continuing the example from above, if dx=0 and dy=0 the mask covers texel groups (3\*8, 4\*8) to (3\*8+7, 4\*8+7) inclusive. If, on the other hand, dx=2, the rightmost 2 columns in the mask have their x coordinates reduced by 8, and similarly for dy.

See the OptiX SDK for sample code that illustrates how to unpack the result.

### 5.1.2.162 optixTexFootprint2DGrad()

```
static __forceinline__ __device__ uint4 optixTexFootprint2DGrad (
    unsigned long long tex,
    unsigned int texInfo,
    float x,
    float y,
    float dPdx_x,
    float dPdx_y,
    float dPdy_x,
    float dPdy_y,
    bool coarse,
    unsigned int * singleMipLevel ) [static]
```

optixTexFootprint2DGrad calculates the footprint of a corresponding 2D texture fetch (tex2DGrad)

#### Parameters

in	<i>tex</i>	CUDA texture object (cast to 64-bit integer)
in	<i>texInfo</i>	Texture info packed into 32-bit integer, described below.
in	<i>x</i>	Texture coordinate
in	<i>y</i>	Texture coordinate
in	<i>dPdx_x</i>	Derivative of x coordinte, which determines level of detail.
in	<i>dPdx_y</i>	Derivative of x coordinte, which determines level of detail.
in	<i>dPdy_x</i>	Derivative of y coordinte, which determines level of detail.



## Parameters

in	<i>dPdy_y</i>	Derivative of y coordinte, which determines level of detail.
in	<i>coarse</i>	Requests footprint from coarse miplevel, when the footprint spans two levels.
out	<i>singleMipLevel</i>	Result indicating whether the footprint spans only a single miplevel.

See also [optixTexFootprint2D\(unsigned long long,unsigned int,float,float,unsigned int\\*\)](#)

## 5.1.2.163 optixTexFootprint2DLod()

```
static __forceinline__ __device__ uint4 optixTexFootprint2DLod (
    unsigned long long tex,
    unsigned int texInfo,
    float x,
    float y,
    float level,
    bool coarse,
    unsigned int * singleMipLevel ) [static]
```

optixTexFootprint2DLod calculates the footprint of a corresponding 2D texture fetch (tex2DLod)

## Parameters

in	<i>tex</i>	CUDA texture object (cast to 64-bit integer)
in	<i>texInfo</i>	Texture info packed into 32-bit integer, described below.
in	<i>x</i>	Texture coordinate
in	<i>y</i>	Texture coordinate
in	<i>level</i>	Level of detail (lod)
in	<i>coarse</i>	Requests footprint from coarse miplevel, when the footprint spans two levels.
out	<i>singleMipLevel</i>	Result indicating whether the footprint spans only a single miplevel.

See also [optixTexFootprint2D\(unsigned long long,unsigned int,float,float,unsigned int\\*\)](#)

## 5.1.2.164 optixThrowException() [1/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode ) [static]
```

Throws a user exception with the given exception code (overload without exception details).

The exception code must be in the range from 0 to  $2^{30} - 1$ . Up to 8 optional exception details can be passed. They can be queried in the EX program using [optixGetExceptionDetail\\_0\(\)](#) to ...[\\_8\(\)](#).

The exception details must not be used to encode pointers to the stack since the current stack is not preserved in the EX program.

Not available in EX.

## Parameters

in	<i>exceptionCode</i>	The exception code to be thrown.
----	----------------------	----------------------------------

5.1.2.165 `optixThrowException()` [2/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode,
    unsigned int exceptionDetail0 ) [static]
```

Throws a user exception with the given exception code (overload with 1 exception detail).

See also [optixThrowException\(int\)](#)

5.1.2.166 `optixThrowException()` [3/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode,
    unsigned int exceptionDetail0,
    unsigned int exceptionDetail1 ) [static]
```

Throws a user exception with the given exception code (overload with 2 exception details).

See also [optixThrowException\(int\)](#)

5.1.2.167 `optixThrowException()` [4/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode,
    unsigned int exceptionDetail0,
    unsigned int exceptionDetail1,
    unsigned int exceptionDetail2 ) [static]
```

Throws a user exception with the given exception code (overload with 3 exception details).

See also [optixThrowException\(int\)](#)

5.1.2.168 `optixThrowException()` [5/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode,
    unsigned int exceptionDetail0,
    unsigned int exceptionDetail1,
    unsigned int exceptionDetail2,
    unsigned int exceptionDetail3 ) [static]
```

Throws a user exception with the given exception code (overload with 4 exception details).

See also [optixThrowException\(int\)](#)

5.1.2.169 `optixThrowException()` [6/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode,
    unsigned int exceptionDetail0,
    unsigned int exceptionDetail1,
    unsigned int exceptionDetail2,
    unsigned int exceptionDetail3,
```

```
    unsigned int exceptionDetail4 ) [static]
```

Throws a user exception with the given exception code (overload with 5 exception details).

See also [optixThrowException\(int\)](#)

#### 5.1.2.170 optixThrowException() [7/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode,
    unsigned int exceptionDetail0,
    unsigned int exceptionDetail1,
    unsigned int exceptionDetail2,
    unsigned int exceptionDetail3,
    unsigned int exceptionDetail4,
    unsigned int exceptionDetail5 ) [static]
```

Throws a user exception with the given exception code (overload with 6 exception details).

See also [optixThrowException\(int\)](#)

#### 5.1.2.171 optixThrowException() [8/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode,
    unsigned int exceptionDetail0,
    unsigned int exceptionDetail1,
    unsigned int exceptionDetail2,
    unsigned int exceptionDetail3,
    unsigned int exceptionDetail4,
    unsigned int exceptionDetail5,
    unsigned int exceptionDetail6 ) [static]
```

Throws a user exception with the given exception code (overload with 7 exception details).

See also [optixThrowException\(int\)](#)

#### 5.1.2.172 optixThrowException() [9/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode,
    unsigned int exceptionDetail0,
    unsigned int exceptionDetail1,
    unsigned int exceptionDetail2,
    unsigned int exceptionDetail3,
    unsigned int exceptionDetail4,
    unsigned int exceptionDetail5,
    unsigned int exceptionDetail6,
    unsigned int exceptionDetail7 ) [static]
```

Throws a user exception with the given exception code (overload with 8 exception details).

See also [optixThrowException\(int\)](#)

### 5.1.2.173 optixTrace() [1/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixTrace (
    OptixPayloadTypeID type,
    OptixTraversableHandle handle,
    float3 rayOrigin,
    float3 rayDirection,
    float tmin,
    float tmax,
    float rayTime,
    OptixVisibilityMask visibilityMask,
    unsigned int rayFlags,
    unsigned int SBTOffset,
    unsigned int SBTstride,
    unsigned int missSBTIndex,
    Payload &... payload ) [static]
```

Initiates a ray tracing query starting with the given traversable.

Parameters

in	<i>type</i>	
in	<i>handle</i>	
in	<i>rayOrigin</i>	
in	<i>rayDirection</i>	
in	<i>tmin</i>	
in	<i>tmax</i>	
in	<i>rayTime</i>	
in	<i>visibilityMask</i>	really only 8 bits
in	<i>rayFlags</i>	really only 16 bits, combination of OptixRayFlags
in	<i>SBTOffset</i>	really only 4 bits
in	<i>SBTstride</i>	really only 4 bits
in	<i>missSBTIndex</i>	specifies the miss program invoked on a miss
in, out	<i>payload</i>	up to 32 unsigned int values that hold the payload

### 5.1.2.174 optixTrace() [2/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixTrace (
    OptixTraversableHandle handle,
    float3 rayOrigin,
```

```

float3 rayDirection,
float tmin,
float tmax,
float rayTime,
OptixVisibilityMask visibilityMask,
unsigned int rayFlags,
unsigned int SBTOffset,
unsigned int SBTstride,
unsigned int missSBTIndex,
Payload &... payload ) [static]

```

Initiates a ray tracing query starting with the given traversable.

Parameters

in	<i>handle</i>	
in	<i>rayOrigin</i>	
in	<i>rayDirection</i>	
in	<i>tmin</i>	
in	<i>tmax</i>	
in	<i>rayTime</i>	
in	<i>visibilityMask</i>	really only 8 bits
in	<i>rayFlags</i>	really only 16 bits, combination of OptixRayFlags
in	<i>SBTOffset</i>	really only 4 bits
in	<i>SBTstride</i>	really only 4 bits
in	<i>missSBTIndex</i>	specifies the miss program invoked on a miss
in, out	<i>payload</i>	up to 32 unsigned int values that hold the payload

#### 5.1.2.175 optixTransformNormalFromObjectToWorldSpace()

```

static __forceinline__ __device__ float3
optixTransformNormalFromObjectToWorldSpace (
    float3 normal ) [static]

```

Transforms the normal using object-to-world transformation matrix resulting from the current active transformation list.

The cost of this function may be proportional to the size of the transformation list.

#### 5.1.2.176 optixTransformNormalFromWorldToObjectSpace()

```

static __forceinline__ __device__ float3
optixTransformNormalFromWorldToObjectSpace (
    float3 normal ) [static]

```

Transforms the normal using world-to-object transformation matrix resulting from the current active transformation list.

The cost of this function may be proportional to the size of the transformation list.

### 5.1.2.177 optixTransformPointFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixTransformPointFromObjectToWorldSpace (
    float3 point ) [static]
```

Transforms the point using object-to-world transformation matrix resulting from the current active transformation list.

The cost of this function may be proportional to the size of the transformation list.

### 5.1.2.178 optixTransformPointFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixTransformPointFromWorldToObjectSpace (
    float3 point ) [static]
```

Transforms the point using world-to-object transformation matrix resulting from the current active transformation list.

The cost of this function may be proportional to the size of the transformation list.

### 5.1.2.179 optixTransformVectorFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixTransformVectorFromObjectToWorldSpace (
    float3 vec ) [static]
```

Transforms the vector using object-to-world transformation matrix resulting from the current active transformation list.

The cost of this function may be proportional to the size of the transformation list.

### 5.1.2.180 optixTransformVectorFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixTransformVectorFromWorldToObjectSpace (
    float3 vec ) [static]
```

Transforms the vector using world-to-object transformation matrix resulting from the current active transformation list.

The cost of this function may be proportional to the size of the transformation list.

### 5.1.2.181 optixUndefinedValue()

```
static __forceinline__ __device__ unsigned int optixUndefinedValue ( ) [static]
```

Returns an undefined value.

## 5.2 Function Table

### Classes

- struct [OptixFunctionTable](#)

### Typedefs

- typedef struct [OptixFunctionTable](#) [OptixFunctionTable](#)

## Variables

- [OptixFunctionTable g\\_optixFunctionTable](#)

### 5.2.1 Detailed Description

OptiX Function Table.

### 5.2.2 Typedef Documentation

#### 5.2.2.1 OptixFunctionTable

```
typedef struct OptixFunctionTable OptixFunctionTable
```

The function table containing all API functions.

See [optixInit\(\)](#) and [optixInitWithHandle\(\)](#).

### 5.2.3 Variable Documentation

#### 5.2.3.1 g\_optixFunctionTable

[OptixFunctionTable](#) [g\\_optixFunctionTable](#)

If the stubs in [optix\\_stubs.h](#) are used, then the function table needs to be defined in exactly one translation unit. This can be achieved by including this header file in that translation unit.

## 5.3 Host API

### Modules

- [Error handling](#)
- [Device context](#)
- [Pipelines](#)
- [Modules](#)
- [Tasks](#)
- [Program groups](#)
- [Launches](#)
- [Acceleration structures](#)
- [Denoiser](#)

### 5.3.1 Detailed Description

OptiX Host API.

## 5.4 Error handling

## 5.5 Device context

## 5.6 Pipelines

## 5.7 Modules

## 5.8 Tasks

## 5.9 Program groups

## 5.10 Launches

## 5.11 Acceleration structures

## 5.12 Denoiser

## 5.13 Utilities

### Classes

- struct [OptixUtilDenoiserImageTile](#)

### Functions

- [OptixResult optixUtilGetPixelStride](#) (const [OptixImage2D](#) &image, unsigned int &pixelStrideInBytes)
- [OptixResult optixUtilDenoiserSplitImage](#) (const [OptixImage2D](#) &input, const [OptixImage2D](#) &output, unsigned int overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight, std::vector< [OptixUtilDenoiserImageTile](#) > &tiles)
- [OptixResult optixUtilDenoiserInvokeTiled](#) ([OptixDenoiser](#) denoiser, [CUstream](#) stream, const [OptixDenoiserParams](#) \*params, [CUdeviceptr](#) denoiserState, size\_t denoiserStateSizeInBytes, const [OptixDenoiserGuideLayer](#) \*guideLayer, const [OptixDenoiserLayer](#) \*layers, unsigned int numLayers, [CUdeviceptr](#) scratch, size\_t scratchSizeInBytes, unsigned int overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight)
- [OptixResult optixUtilAccumulateStackSizes](#) ([OptixProgramGroup](#) programGroup, [OptixStackSizes](#) \*stackSizes, [OptixPipeline](#) pipeline)
- [OptixResult optixUtilComputeStackSizes](#) (const [OptixStackSizes](#) \*stackSizes, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepth, unsigned int \*directCallableStackSizeFromTraversal, unsigned int \*directCallableStackSizeFromState, unsigned int \*continuationStackSize)
- [OptixResult optixUtilComputeStackSizesDCSplit](#) (const [OptixStackSizes](#) \*stackSizes, unsigned int dssDCFromTraversal, unsigned int dssDCFromState, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepthFromTraversal, unsigned int maxDCDepthFromState, unsigned int \*directCallableStackSizeFromTraversal, unsigned int \*directCallableStackSizeFromState, unsigned int \*continuationStackSize)
- [OptixResult optixUtilComputeStackSizesCssCCTree](#) (const [OptixStackSizes](#) \*stackSizes, unsigned int cssCCTree, unsigned int maxTraceDepth, unsigned int maxDCDepth, unsigned int \*directCallableStackSizeFromTraversal, unsigned int \*directCallableStackSizeFromState, unsigned int \*continuationStackSize)
- [OptixResult optixUtilComputeStackSizesSimplePathTracer](#) ([OptixProgramGroup](#) programGroupRG, [OptixProgramGroup](#) programGroupMS1, const [OptixProgramGroup](#) \*programGroupCH1, unsigned int programGroupCH1Count, [OptixProgramGroup](#)



```
programGroupMS2, const OptixProgramGroup *programGroupCH2, unsigned int
programGroupCH2Count, unsigned int *directCallableStackSizeFromTraversal, unsigned int
*directCallableStackSizeFromState, unsigned int *continuationStackSize, OptixPipeline pipeline)
```

- [OptixResult](#) [optixInitWithHandle](#) (void \*\*handlePtr)
- [OptixResult](#) [optixInit](#) (void)
- [OptixResult](#) [optixUninitWithHandle](#) (void \*handle)

### 5.13.1 Detailed Description

OptiX Utilities.

### 5.13.2 Function Documentation

#### 5.13.2.1 [optixInit](#)()

```
OptixResult optixInit (
    void ) [inline]
```

Loads the OptiX library and initializes the function table used by the stubs below.

A variant of [optixInitWithHandle](#)() that does not make the handle to the loaded library available.

#### 5.13.2.2 [optixInitWithHandle](#)()

```
OptixResult optixInitWithHandle (
    void ** handlePtr ) [inline]
```

Loads the OptiX library and initializes the function table used by the stubs below.

If handlePtr is not nullptr, an OS-specific handle to the library will be returned in \*handlePtr.

See also [optixUninitWithHandle](#)

#### 5.13.2.3 [optixUninitWithHandle](#)()

```
OptixResult optixUninitWithHandle (
    void * handle ) [inline]
```

Unloads the OptiX library and zeros the function table used by the stubs below. Takes the handle returned by [optixInitWithHandle](#). All [OptixDeviceContext](#) objects must be destroyed before calling this function, or the behavior is undefined.

See also [optixInitWithHandle](#)

#### 5.13.2.4 [optixUtilAccumulateStackSizes](#)()

```
OptixResult optixUtilAccumulateStackSizes (
    OptixProgramGroup programGroup,
    OptixStackSizes * stackSizes,
    OptixPipeline pipeline ) [inline]
```

Retrieves direct and continuation stack sizes for each program in the program group and accumulates the upper bounds in the corresponding output variables based on the semantic type of the program. Before the first invocation of this function with a given instance of [OptixStackSizes](#), the members of that instance should be set to 0. If the programs rely on external functions, passing the current pipeline will consider these as well. Otherwise, a null pointer can be passed instead. When external functions are present, a warning will be issued for these cases.

### 5.13.2.5 optixUtilComputeStackSizes()

```
OptixResult optixUtilComputeStackSizes (
    const OptixStackSizes * stackSizes,
    unsigned int maxTraceDepth,
    unsigned int maxCCDepth,
    unsigned int maxDCDepth,
    unsigned int * directCallableStackSizeFromTraversal,
    unsigned int * directCallableStackSizeFromState,
    unsigned int * continuationStackSize ) [inline]
```

Computes the stack size values needed to configure a pipeline.

See the programming guide for an explanation of the formula.

#### Parameters

in	<i>stackSizes</i>	Accumulated stack sizes of all programs in the call graph.
in	<i>maxTraceDepth</i>	Maximum depth of <code>optixTrace()</code> calls.
in	<i>maxCCDepth</i>	Maximum depth of calls trees of continuation callables.
in	<i>maxDCDepth</i>	Maximum depth of calls trees of direct callables.
out	<i>directCallableStackSizeFromTraversal</i>	Direct stack size requirement for direct callables invoked from IS or AH.
out	<i>directCallableStackSizeFromState</i>	Direct stack size requirement for direct callables invoked from RG, MS, or CH.
out	<i>continuationStackSize</i>	Continuation stack requirement.

### 5.13.2.6 optixUtilComputeStackSizesCssCCTree()

```
OptixResult optixUtilComputeStackSizesCssCCTree (
    const OptixStackSizes * stackSizes,
    unsigned int cssCCTree,
    unsigned int maxTraceDepth,
    unsigned int maxDCDepth,
    unsigned int * directCallableStackSizeFromTraversal,
    unsigned int * directCallableStackSizeFromState,
    unsigned int * continuationStackSize ) [inline]
```

Computes the stack size values needed to configure a pipeline.

This variant is similar to `optixUtilComputeStackSizes()`, except that it expects the value `cssCCTree` instead of `cssCC` and `maxCCDepth`.

See programming guide for an explanation of the formula.

#### Parameters

in	<i>stackSizes</i>	Accumulated stack sizes of all programs in the call graph.
----	-------------------	--

## Parameters

in	<i>cssCCTree</i>	Maximum stack size used by calls trees of continuation callables.
in	<i>maxTraceDepth</i>	Maximum depth of <code>optixTrace()</code> calls.
in	<i>maxDCDepth</i>	Maximum depth of calls trees of direct callables.
out	<i>directCallableStackSizeFromTraversal</i>	Direct stack size requirement for direct callables invoked from IS or AH.
out	<i>directCallableStackSizeFromState</i>	Direct stack size requirement for direct callables invoked from RG, MS, or CH.
out	<i>continuationStackSize</i>	Continuation stack requirement.

5.13.2.7 `optixUtilComputeStackSizesDCSplit()`

```

OptixResult optixUtilComputeStackSizesDCSplit (
    const OptixStackSizes * stackSizes,
    unsigned int dssDCFromTraversal,
    unsigned int dssDCFromState,
    unsigned int maxTraceDepth,
    unsigned int maxCCDepth,
    unsigned int maxDCDepthFromTraversal,
    unsigned int maxDCDepthFromState,
    unsigned int * directCallableStackSizeFromTraversal,
    unsigned int * directCallableStackSizeFromState,
    unsigned int * continuationStackSize ) [inline]

```

Computes the stack size values needed to configure a pipeline.

This variant is similar to `optixUtilComputeStackSizes()`, except that it expects the values `dssDC` and `maxDCDepth` split by call site semantic.

See programming guide for an explanation of the formula.

## Parameters

in	<i>stackSizes</i>	Accumulated stack sizes of all programs in the call graph.
in	<i>dssDCFromTraversal</i>	Accumulated direct stack size of all DC programs invoked from IS or AH.
in	<i>dssDCFromState</i>	Accumulated direct stack size of all DC programs invoked from RG, MS, or CH.
in	<i>maxTraceDepth</i>	Maximum depth of <code>optixTrace()</code> calls.
in	<i>maxCCDepth</i>	Maximum depth of calls trees of continuation callables.
in	<i>maxDCDepthFromTraversal</i>	Maximum depth of calls trees of direct callables invoked from IS or AH.
in	<i>maxDCDepthFromState</i>	Maximum depth of calls trees of direct callables invoked from RG, MS, or CH.

## Parameters

out	<i>directCallableStackSizeFromTraversal</i>	Direct stack size requirement for direct callables invoked from IS or AH.
out	<i>directCallableStackSizeFromState</i>	Direct stack size requirement for direct callables invoked from RG, MS, or CH.
out	<i>continuationStackSize</i>	Continuation stack requirement.

## 5.13.2.8 optixUtilComputeStackSizesSimplePathTracer()

```

OptixResult optixUtilComputeStackSizesSimplePathTracer (
    OptixProgramGroup programGroupRG,
    OptixProgramGroup programGroupMS1,
    const OptixProgramGroup * programGroupCH1,
    unsigned int programGroupCH1Count,
    OptixProgramGroup programGroupMS2,
    const OptixProgramGroup * programGroupCH2,
    unsigned int programGroupCH2Count,
    unsigned int * directCallableStackSizeFromTraversal,
    unsigned int * directCallableStackSizeFromState,
    unsigned int * continuationStackSize,
    OptixPipeline pipeline ) [inline]

```

Computes the stack size values needed to configure a pipeline.

This variant is a specialization of `optixUtilComputeStackSizes()` for a simple path tracer with the following assumptions: There are only two ray types, camera rays and shadow rays. There are only RG, MS, and CH programs, and no AH, IS, CC, or DC programs. The camera rays invoke only the miss and closest hit programs MS1 and CH1, respectively. The CH1 program might trace shadow rays, which invoke only the miss and closest hit programs MS2 and CH2, respectively.

For flexibility, we allow for each of CH1 and CH2 not just one single program group, but an array of programs groups, and compute the maximas of the stack size requirements per array.

See programming guide for an explanation of the formula.

If the programs rely on external functions, passing the current pipeline will consider these as well. Otherwise, a null pointer can be passed instead. When external functions are present, a warning will be issued for these cases.

## 5.13.2.9 optixUtilDenoiserInvokeTiled()

```

OptixResult optixUtilDenoiserInvokeTiled (
    OptixDenoiser denoiser,
    CUstream stream,
    const OptixDenoiserParams * params,
    CUdeviceptr denoiserState,
    size_t denoiserStateSizeInBytes,
    const OptixDenoiserGuideLayer * guideLayer,
    const OptixDenoiserLayer * layers,

```

```

    unsigned int numLayers,
    CUdeviceptr scratch,
    size_t scratchSizeInBytes,
    unsigned int overlapWindowSizeInPixels,
    unsigned int tileWidth,
    unsigned int tileHeight ) [inline]

```

Run denoiser on input layers see [optixDenoiserInvoke](#) additional parameters:

Runs the denoiser on the input layers on a single GPU and stream using [optixDenoiserInvoke](#). If the input layers' dimensions are larger than the specified tile size, the image is divided into tiles using [optixUtilDenoiserSplitImage](#), and multiple back-to-back invocations are performed in order to reuse the scratch space. Multiple tiles can be invoked concurrently if [optixUtilDenoiserSplitImage](#) is used directly and multiple scratch allocations for each concurrent invocation are used. The input parameters are the same as [optixDenoiserInvoke](#) except for the addition of the maximum tile size.

#### Parameters

in	<i>denoiser</i>
in	<i>stream</i>
in	<i>params</i>
in	<i>denoiserState</i>
in	<i>denoiserStateSizeInBytes</i>
in	<i>guideLayer</i>
in	<i>layers</i>
in	<i>numLayers</i>
in	<i>scratch</i>
in	<i>scratchSizeInBytes</i>
in	<i>overlapWindowSizeInPixels</i>
in	<i>tileWidth</i>
in	<i>tileHeight</i>

#### 5.13.2.10 optixUtilDenoiserSplitImage()

```

OptixResult optixUtilDenoiserSplitImage (
    const OptixImage2D & input,
    const OptixImage2D & output,
    unsigned int overlapWindowSizeInPixels,
    unsigned int tileWidth,
    unsigned int tileHeight,
    std::vector< OptixUtilDenoiserImageTile > & tiles ) [inline]

```

Split image into 2D tiles given horizontal and vertical tile size.

#### Parameters

in	<i>input</i>	full resolution input image to be split
----	--------------	---

## Parameters

in	<i>output</i>	full resolution output image
in	<i>overlapWindowSizeInPixels</i>	see <a href="#">OptixDenoiserSizes</a> , <a href="#">optixDenoiserComputeMemoryResources</a>
in	<i>tileWidth</i>	maximum width of tiles
in	<i>tileHeight</i>	maximum height of tiles
out	<i>tiles</i>	list of tiles covering the input image

5.13.2.11 `optixUtilGetPixelStride()`

```
OptixResult optixUtilGetPixelStride (
    const OptixImage2D & image,
    unsigned int & pixelStrideInBytes ) [inline]
```

Return pixel stride in bytes for the given pixel format if the `pixelStrideInBytes` member of the image is zero. Otherwise return `pixelStrideInBytes` from the image.

## Parameters

in	<i>image</i>	Image containing the pixel stride
in	<i>pixelStrideInBytes</i>	Pixel stride in bytes

## 5.14 Types

## Classes

- struct [OptixDeviceContextOptions](#)
- struct [OptixOpacityMicromapUsageCount](#)
- struct [OptixBuildInputOpacityMicromap](#)
- struct [OptixRelocateInputOpacityMicromap](#)
- struct [OptixDisplacementMicromapDesc](#)
- struct [OptixDisplacementMicromapHistogramEntry](#)
- struct [OptixDisplacementMicromapArrayBuildInput](#)
- struct [OptixDisplacementMicromapUsageCount](#)
- struct [OptixBuildInputDisplacementMicromap](#)
- struct [OptixBuildInputTriangleArray](#)
- struct [OptixRelocateInputTriangleArray](#)
- struct [OptixBuildInputCurveArray](#)
- struct [OptixBuildInputSphereArray](#)
- struct [OptixAabb](#)
- struct [OptixBuildInputCustomPrimitiveArray](#)
- struct [OptixBuildInputInstanceArray](#)
- struct [OptixRelocateInputInstanceArray](#)
- struct [OptixBuildInput](#)
- struct [OptixRelocateInput](#)
- struct [OptixInstance](#)
- struct [OptixOpacityMicromapDesc](#)
- struct [OptixOpacityMicromapHistogramEntry](#)
- struct [OptixOpacityMicromapArrayBuildInput](#)

- struct OptixMicromapBufferSizes
- struct OptixMicromapBuffers
- struct OptixMotionOptions
- struct OptixAccelBuildOptions
- struct OptixAccelBufferSizes
- struct OptixAccelEmitDesc
- struct OptixRelocationInfo
- struct OptixStaticTransform
- struct OptixMatrixMotionTransform
- struct OptixSRTData
- struct OptixSRTMotionTransform
- struct OptixImage2D
- struct OptixDenoiserOptions
- struct OptixDenoiserGuideLayer
- struct OptixDenoiserLayer
- struct OptixDenoiserParams
- struct OptixDenoiserSizes
- struct OptixModuleCompileBoundValueEntry
- struct OptixPayloadType
- struct OptixModuleCompileOptions
- struct OptixProgramGroupSingleModule
- struct OptixProgramGroupHitgroup
- struct OptixProgramGroupCallables
- struct OptixProgramGroupDesc
- struct OptixProgramGroupOptions
- struct OptixPipelineCompileOptions
- struct OptixPipelineLinkOptions
- struct OptixShaderBindingTable
- struct OptixStackSizes
- struct OptixBuiltinISOOptions

## Macros

- #define OPTIX\_SBT\_RECORD\_HEADER\_SIZE ((size\_t)32)
- #define OPTIX\_SBT\_RECORD\_ALIGNMENT 16ull
- #define OPTIX\_ACCEL\_BUFFER\_BYTE\_ALIGNMENT 128ull
- #define OPTIX\_INSTANCE\_BYTE\_ALIGNMENT 16ull
- #define OPTIX\_AABB\_BUFFER\_BYTE\_ALIGNMENT 8ull
- #define OPTIX\_GEOMETRY\_TRANSFORM\_BYTE\_ALIGNMENT 16ull
- #define OPTIX\_TRANSFORM\_BYTE\_ALIGNMENT 64ull
- #define OPTIX\_OPACITY\_MICROMAP\_DESC\_BUFFER\_BYTE\_ALIGNMENT 8ull
- #define OPTIX\_COMPILE\_DEFAULT\_MAX\_REGISTER\_COUNT 0
- #define OPTIX\_COMPILE\_DEFAULT\_MAX\_PAYLOAD\_TYPE\_COUNT 8
- #define OPTIX\_COMPILE\_DEFAULT\_MAX\_PAYLOAD\_VALUE\_COUNT 32
- #define OPTIX\_OPACITY\_MICROMAP\_STATE\_TRANSPARENT (0)
- #define OPTIX\_OPACITY\_MICROMAP\_STATE\_OPAQUE (1)
- #define OPTIX\_OPACITY\_MICROMAP\_STATE\_UNKNOWN\_TRANSPARENT (2)
- #define OPTIX\_OPACITY\_MICROMAP\_STATE\_UNKNOWN\_OPAQUE (3)
- #define OPTIX\_OPACITY\_MICROMAP\_PREDEFINED\_INDEX\_FULLY\_TRANSPARENT (-1)
- #define OPTIX\_OPACITY\_MICROMAP\_PREDEFINED\_INDEX\_FULLY\_OPAQUE (-2)



- `#define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT (-3)`
- `#define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE (-4)`
- `#define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull`
- `#define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12`
- `#define OPTIX_DISPLACEMENT_MICROMAP_MAX_SUBDIVISION_LEVEL 5`
- `#define OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull`
- `#define OPTIX_DISPLACEMENT_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull`

## Typedefs

- `typedef unsigned long long CUdeviceptr`
- `typedef struct OptixDeviceContext_t * OptixDeviceContext`
- `typedef struct OptixModule_t * OptixModule`
- `typedef struct OptixProgramGroup_t * OptixProgramGroup`
- `typedef struct OptixPipeline_t * OptixPipeline`
- `typedef struct OptixDenoiser_t * OptixDenoiser`
- `typedef struct OptixTask_t * OptixTask`
- `typedef unsigned long long OptixTraversableHandle`
- `typedef unsigned int OptixVisibilityMask`
- `typedef enum OptixResult OptixResult`
- `typedef enum OptixDeviceProperty OptixDeviceProperty`
- `typedef void(* OptixLogCallback) (unsigned int level, const char *tag, const char *message, void *cbdata)`
- `typedef enum OptixDeviceContextValidationMode OptixDeviceContextValidationMode`
- `typedef struct OptixDeviceContextOptions OptixDeviceContextOptions`
- `typedef enum OptixGeometryFlags OptixGeometryFlags`
- `typedef enum OptixHitKind OptixHitKind`
- `typedef enum OptixIndicesFormat OptixIndicesFormat`
- `typedef enum OptixVertexFormat OptixVertexFormat`
- `typedef enum OptixTransformFormat OptixTransformFormat`
- `typedef enum OptixDisplacementMicromapBiasAndScaleFormat OptixDisplacementMicromapBiasAndScaleFormat`
- `typedef enum OptixDisplacementMicromapDirectionFormat OptixDisplacementMicromapDirectionFormat`
- `typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat`
- `typedef enum OptixOpacityMicromapArrayIndexingMode OptixOpacityMicromapArrayIndexingMode`
- `typedef struct OptixOpacityMicromapUsageCount OptixOpacityMicromapUsageCount`
- `typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap`
- `typedef struct OptixRelocateInputOpacityMicromap OptixRelocateInputOpacityMicromap`
- `typedef enum OptixDisplacementMicromapFormat OptixDisplacementMicromapFormat`
- `typedef enum OptixDisplacementMicromapFlags OptixDisplacementMicromapFlags`
- `typedef enum OptixDisplacementMicromapTriangleFlags OptixDisplacementMicromapTriangleFlags`
- `typedef struct OptixDisplacementMicromapDesc OptixDisplacementMicromapDesc`
- `typedef struct OptixDisplacementMicromapHistogramEntry OptixDisplacementMicromapHistogramEntry`
- `typedef struct OptixDisplacementMicromapArrayBuildInput OptixDisplacementMicromapArrayBuildInput`



- typedef struct OptixDisplacementMicromapUsageCount  
OptixDisplacementMicromapUsageCount
- typedef enum OptixDisplacementMicromapArrayIndexingMode  
OptixDisplacementMicromapArrayIndexingMode
- typedef struct OptixBuildInputDisplacementMicromap OptixBuildInputDisplacementMicromap
- typedef struct OptixBuildInputTriangleArray OptixBuildInputTriangleArray
- typedef struct OptixRelocateInputTriangleArray OptixRelocateInputTriangleArray
- typedef enum OptixPrimitiveType OptixPrimitiveType
- typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags
- typedef enum OptixCurveEndcapFlags OptixCurveEndcapFlags
- typedef struct OptixBuildInputCurveArray OptixBuildInputCurveArray
- typedef struct OptixBuildInputSphereArray OptixBuildInputSphereArray
- typedef struct OptixAabb OptixAabb
- typedef struct OptixBuildInputCustomPrimitiveArray OptixBuildInputCustomPrimitiveArray
- typedef struct OptixBuildInputInstanceArray OptixBuildInputInstanceArray
- typedef struct OptixRelocateInputInstanceArray OptixRelocateInputInstanceArray
- typedef enum OptixBuildInputType OptixBuildInputType
- typedef struct OptixBuildInput OptixBuildInput
- typedef struct OptixRelocateInput OptixRelocateInput
- typedef enum OptixInstanceFlags OptixInstanceFlags
- typedef struct OptixInstance OptixInstance
- typedef enum OptixBuildFlags OptixBuildFlags
- typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags
- typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc
- typedef struct OptixOpacityMicromapHistogramEntry OptixOpacityMicromapHistogramEntry
- typedef struct OptixOpacityMicromapArrayBuildInput OptixOpacityMicromapArrayBuildInput
- typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
- typedef struct OptixMicromapBuffers OptixMicromapBuffers
- typedef enum OptixBuildOperation OptixBuildOperation
- typedef enum OptixMotionFlags OptixMotionFlags
- typedef struct OptixMotionOptions OptixMotionOptions
- typedef struct OptixAccelBuildOptions OptixAccelBuildOptions
- typedef struct OptixAccelBufferSizes OptixAccelBufferSizes
- typedef enum OptixAccelPropertyType OptixAccelPropertyType
- typedef struct OptixAccelEmitDesc OptixAccelEmitDesc
- typedef struct OptixRelocationInfo OptixRelocationInfo
- typedef struct OptixStaticTransform OptixStaticTransform
- typedef struct OptixMatrixMotionTransform OptixMatrixMotionTransform
- typedef struct OptixSRTData OptixSRTData
- typedef struct OptixSRTMotionTransform OptixSRTMotionTransform
- typedef enum OptixTraversableType OptixTraversableType
- typedef enum OptixPixelFormat OptixPixelFormat
- typedef struct OptixImage2D OptixImage2D
- typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
- typedef struct OptixDenoiserOptions OptixDenoiserOptions
- typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
- typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
- typedef struct OptixDenoiserLayer OptixDenoiserLayer
- typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
- typedef struct OptixDenoiserParams OptixDenoiserParams

- typedef struct OptixDenoiserSizes OptixDenoiserSizes
- typedef enum OptixRayFlags OptixRayFlags
- typedef enum OptixTransformType OptixTransformType
- typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags
- typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
- typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
- typedef enum OptixModuleCompileState OptixModuleCompileState
- typedef struct OptixModuleCompileBoundValueEntry OptixModuleCompileBoundValueEntry
- typedef enum OptixPayloadTypeID OptixPayloadTypeID
- typedef enum OptixPayloadSemantics OptixPayloadSemantics
- typedef struct OptixPayloadType OptixPayloadType
- typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
- typedef enum OptixProgramGroupKind OptixProgramGroupKind
- typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
- typedef struct OptixProgramGroupSingleModule OptixProgramGroupSingleModule
- typedef struct OptixProgramGroupHitgroup OptixProgramGroupHitgroup
- typedef struct OptixProgramGroupCallables OptixProgramGroupCallables
- typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
- typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
- typedef enum OptixExceptionCodes OptixExceptionCodes
- typedef enum OptixExceptionFlags OptixExceptionFlags
- typedef struct OptixPipelineCompileOptions OptixPipelineCompileOptions
- typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
- typedef struct OptixShaderBindingTable OptixShaderBindingTable
- typedef struct OptixStackSizes OptixStackSizes
- typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
- typedef OptixResult() OptixQueryFunctionTable\_t(int abiId, unsigned int numOptions, OptixQueryFunctionTableOptions \*, const void \*\*, void \*functionTable, size\_t sizeOfTable)
- typedef struct OptixBuiltinISOOptions OptixBuiltinISOOptions

## Enumerations

- enum OptixResult {  
OPTIX\_SUCCESS = 0 ,  
OPTIX\_ERROR\_INVALID\_VALUE = 7001 ,  
OPTIX\_ERROR\_HOST\_OUT\_OF\_MEMORY = 7002 ,  
OPTIX\_ERROR\_INVALID\_OPERATION = 7003 ,  
OPTIX\_ERROR\_FILE\_IO\_ERROR = 7004 ,  
OPTIX\_ERROR\_INVALID\_FILE\_FORMAT = 7005 ,  
OPTIX\_ERROR\_DISK\_CACHE\_INVALID\_PATH = 7010 ,  
OPTIX\_ERROR\_DISK\_CACHE\_PERMISSION\_ERROR = 7011 ,  
OPTIX\_ERROR\_DISK\_CACHE\_DATABASE\_ERROR = 7012 ,  
OPTIX\_ERROR\_DISK\_CACHE\_INVALID\_DATA = 7013 ,  
OPTIX\_ERROR\_LAUNCH\_FAILURE = 7050 ,  
OPTIX\_ERROR\_INVALID\_DEVICE\_CONTEXT = 7051 ,  
OPTIX\_ERROR\_CUDA\_NOT\_INITIALIZED = 7052 ,  
OPTIX\_ERROR\_VALIDATION\_FAILURE = 7053 ,  
OPTIX\_ERROR\_INVALID\_INPUT = 7200 ,  
OPTIX\_ERROR\_INVALID\_LAUNCH\_PARAMETER = 7201 ,  
OPTIX\_ERROR\_INVALID\_PAYLOAD\_ACCESS = 7202 ,  
OPTIX\_ERROR\_INVALID\_ATTRIBUTE\_ACCESS = 7203 ,  
OPTIX\_ERROR\_INVALID\_FUNCTION\_USE = 7204 ,

```

OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS = 7205 ,
OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250 ,
OPTIX_ERROR_PIPELINE_LINK_ERROR = 7251 ,
OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE = 7270 ,
OPTIX_ERROR_INTERNAL_COMPILER_ERROR = 7299 ,
OPTIX_ERROR_DENOISER_MODEL_NOT_SET = 7300 ,
OPTIX_ERROR_DENOISER_NOT_INITIALIZED = 7301 ,
OPTIX_ERROR_NOT_COMPATIBLE = 7400 ,
OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH = 7500 ,
OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED = 7501 ,
OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID = 7502 ,
OPTIX_ERROR_NOT_SUPPORTED = 7800 ,
OPTIX_ERROR_UNSUPPORTED_ABI_VERSION = 7801 ,
OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH = 7802 ,
OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS = 7803 ,
OPTIX_ERROR_LIBRARY_NOT_FOUND = 7804 ,
OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND = 7805 ,
OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE = 7806 ,
OPTIX_ERROR_DEVICE_OUT_OF_MEMORY = 7807 ,
OPTIX_ERROR_CUDA_ERROR = 7900 ,
OPTIX_ERROR_INTERNAL_ERROR = 7990 ,
OPTIX_ERROR_UNKNOWN = 7999 }

• enum OptixDeviceProperty {
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004 ,
    OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009 }

• enum OptixDeviceContextValidationMode {
    OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0 ,
    OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF }

• enum OptixGeometryFlags {
    OPTIX_GEOMETRY_FLAG_NONE = 0 ,
    OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
    OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u << 1 ,
    OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 2 }

• enum OptixHitKind {
    OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE ,
    OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF }

• enum OptixIndicesFormat {
    OPTIX_INDICES_FORMAT_NONE = 0 ,
    OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102 ,
    OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103 }

• enum OptixVertexFormat {
    OPTIX_VERTEX_FORMAT_NONE = 0 ,
    OPTIX_VERTEX_FORMAT_FLOAT3 = 0x2121 ,
    OPTIX_VERTEX_FORMAT_FLOAT2 = 0x2122 ,
    OPTIX_VERTEX_FORMAT_HALF3 = 0x2123 ,
    OPTIX_VERTEX_FORMAT_HALF2 = 0x2124 ,

```

- ```

OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125 ,
OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126 }

```
- enum OptixTransformFormat {  
OPTIX\_TRANSFORM\_FORMAT\_NONE = 0 ,  
OPTIX\_TRANSFORM\_FORMAT\_MATRIX\_FLOAT12 = 0x21E1 }
  - enum OptixDisplacementMicromapBiasAndScaleFormat {  
OPTIX\_DISPLACEMENT\_MICROMAP\_BIAS\_AND\_SCALE\_FORMAT\_NONE = 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_BIAS\_AND\_SCALE\_FORMAT\_FLOAT2 = 0x2241 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_BIAS\_AND\_SCALE\_FORMAT\_HALF2 = 0x2242 }
  - enum OptixDisplacementMicromapDirectionFormat {  
OPTIX\_DISPLACEMENT\_MICROMAP\_DIRECTION\_FORMAT\_NONE = 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_DIRECTION\_FORMAT\_FLOAT3 = 0x2261 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_DIRECTION\_FORMAT\_HALF3 = 0x2262 }
  - enum OptixOpacityMicromapFormat {  
OPTIX\_OPACITY\_MICROMAP\_FORMAT\_NONE = 0 ,  
OPTIX\_OPACITY\_MICROMAP\_FORMAT\_2\_STATE = 1 ,  
OPTIX\_OPACITY\_MICROMAP\_FORMAT\_4\_STATE = 2 }
  - enum OptixOpacityMicromapArrayIndexingMode {  
OPTIX\_OPACITY\_MICROMAP\_ARRAY\_INDEXING\_MODE\_NONE = 0 ,  
OPTIX\_OPACITY\_MICROMAP\_ARRAY\_INDEXING\_MODE\_LINEAR = 1 ,  
OPTIX\_OPACITY\_MICROMAP\_ARRAY\_INDEXING\_MODE\_INDEXED = 2 }
  - enum OptixDisplacementMicromapFormat {  
OPTIX\_DISPLACEMENT\_MICROMAP\_FORMAT\_NONE = 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_FORMAT\_64\_MICRO\_TRIS\_64\_BYTES = 1 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_FORMAT\_256\_MICRO\_TRIS\_128\_BYTES = 2 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_FORMAT\_1024\_MICRO\_TRIS\_128\_BYTES = 3 }
  - enum OptixDisplacementMicromapFlags {  
OPTIX\_DISPLACEMENT\_MICROMAP\_FLAG\_NONE = 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_FLAG\_PREFER\_FAST\_TRACE = 1 << 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_FLAG\_PREFER\_FAST\_BUILD = 1 << 1 }
  - enum OptixDisplacementMicromapTriangleFlags {  
OPTIX\_DISPLACEMENT\_MICROMAP\_TRIANGLE\_FLAG\_NONE = 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_TRIANGLE\_FLAG\_DECIMATE\_EDGE\_01 = 1 << 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_TRIANGLE\_FLAG\_DECIMATE\_EDGE\_12 = 1 << 1 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_TRIANGLE\_FLAG\_DECIMATE\_EDGE\_20 = 1 << 2 }
  - enum OptixDisplacementMicromapArrayIndexingMode {  
OPTIX\_DISPLACEMENT\_MICROMAP\_ARRAY\_INDEXING\_MODE\_NONE = 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_ARRAY\_INDEXING\_MODE\_LINEAR = 1 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_ARRAY\_INDEXING\_MODE\_INDEXED = 2 }
  - enum OptixPrimitiveType {  
OPTIX\_PRIMITIVE\_TYPE\_CUSTOM = 0x2500 ,  
OPTIX\_PRIMITIVE\_TYPE\_ROUND\_QUADRATIC\_BSPLINE = 0x2501 ,  
OPTIX\_PRIMITIVE\_TYPE\_ROUND\_CUBIC\_BSPLINE = 0x2502 ,  
OPTIX\_PRIMITIVE\_TYPE\_ROUND\_LINEAR = 0x2503 ,  
OPTIX\_PRIMITIVE\_TYPE\_ROUND\_CATMULLROM = 0x2504 ,  
OPTIX\_PRIMITIVE\_TYPE\_FLAT\_QUADRATIC\_BSPLINE = 0x2505 ,  
OPTIX\_PRIMITIVE\_TYPE\_SPHERE = 0x2506 ,  
OPTIX\_PRIMITIVE\_TYPE\_ROUND\_CUBIC\_BEZIER = 0x2507 ,  
OPTIX\_PRIMITIVE\_TYPE\_TRIANGLE = 0x2531 ,  
OPTIX\_PRIMITIVE\_TYPE\_DISPLACED\_MICROMESH\_TRIANGLE = 0x2532 }
  - enum OptixPrimitiveTypeFlags {  
OPTIX\_PRIMITIVE\_TYPE\_FLAGS\_CUSTOM = 1 << 0 ,

```

OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE = 1 << 1,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE = 1 << 2,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR = 1 << 3,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM = 1 << 4,
OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE = 1 << 5,
OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE = 1 << 6,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER = 1 << 7,
OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE = 1 << 31,
OPTIX_PRIMITIVE_TYPE_FLAGS_DISPLACED_MICROMESH_TRIANGLE = 1 << 30 }
• enum OptixCurveEndcapFlags {
    OPTIX_CURVE_ENDCAP_DEFAULT = 0,
    OPTIX_CURVE_ENDCAP_ON = 1 << 0 }
• enum OptixBuildInputType {
    OPTIX_BUILD_INPUT_TYPE_TRIANGLES = 0x2141,
    OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES = 0x2142,
    OPTIX_BUILD_INPUT_TYPE_INSTANCES = 0x2143,
    OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS = 0x2144,
    OPTIX_BUILD_INPUT_TYPE_CURVES = 0x2145,
    OPTIX_BUILD_INPUT_TYPE_SPHERES = 0x2146 }
• enum OptixInstanceFlags {
    OPTIX_INSTANCE_FLAG_NONE = 0,
    OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 0,
    OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u << 1,
    OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT = 1u << 2,
    OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u << 3,
    OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 4,
    OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS = 1u << 5 }
• enum OptixBuildFlags {
    OPTIX_BUILD_FLAG_NONE = 0,
    OPTIX_BUILD_FLAG_ALLOW_UPDATE = 1u << 0,
    OPTIX_BUILD_FLAG_ALLOW_COMPACTION = 1u << 1,
    OPTIX_BUILD_FLAG_PREFER_FAST_TRACE = 1u << 2,
    OPTIX_BUILD_FLAG_PREFER_FAST_BUILD = 1u << 3,
    OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS = 1u << 4,
    OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS = 1u << 5,
    OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE = 1u << 6,
    OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1u << 7 }
• enum OptixOpacityMicromapFlags {
    OPTIX_OPACITY_MICROMAP_FLAG_NONE = 0,
    OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 << 0,
    OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 << 1 }
• enum OptixBuildOperation {
    OPTIX_BUILD_OPERATION_BUILD = 0x2161,
    OPTIX_BUILD_OPERATION_UPDATE = 0x2162 }
• enum OptixMotionFlags {
    OPTIX_MOTION_FLAG_NONE = 0,
    OPTIX_MOTION_FLAG_START_VANISH = 1u << 0,
    OPTIX_MOTION_FLAG_END_VANISH = 1u << 1 }
• enum OptixAccelPropertyType {
    OPTIX_PROPERTY_TYPE_COMPACTED_SIZE = 0x2181,
    OPTIX_PROPERTY_TYPE_AABBS = 0x2182 }
• enum OptixTraversableType {
    OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM = 0x21C1,

```



```

OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM = 0x21C2 ,
OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM = 0x21C3 }

• enum OptixPixelFormat {
    OPTIX_PIXEL_FORMAT_HALF1 = 0x220a ,
    OPTIX_PIXEL_FORMAT_HALF2 = 0x2207 ,
    OPTIX_PIXEL_FORMAT_HALF3 = 0x2201 ,
    OPTIX_PIXEL_FORMAT_HALF4 = 0x2202 ,
    OPTIX_PIXEL_FORMAT_FLOAT1 = 0x220b ,
    OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208 ,
    OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203 ,
    OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204 ,
    OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205 ,
    OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206 ,
    OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209 }

• enum OptixDenoiserModelKind {
    OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322 ,
    OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323 ,
    OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324 ,
    OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325 ,
    OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326 ,
    OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327 ,
    OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328 }

• enum OptixDenoiserAOVType {
    OPTIX_DENOISER_AOV_TYPE_NONE = 0 ,
    OPTIX_DENOISER_AOV_TYPE_BEAUTY = 0x7000 ,
    OPTIX_DENOISER_AOV_TYPE_SPECULAR = 0x7001 ,
    OPTIX_DENOISER_AOV_TYPE_REFLECTION = 0x7002 ,
    OPTIX_DENOISER_AOV_TYPE_REFRACTION = 0x7003 ,
    OPTIX_DENOISER_AOV_TYPE_DIFFUSE = 0x7004 }

• enum OptixDenoiserAlphaMode {
    OPTIX_DENOISER_ALPHA_MODE_COPY = 0 ,
    OPTIX_DENOISER_ALPHA_MODE_ALPHA_AS_AOV = 1 ,
    OPTIX_DENOISER_ALPHA_MODE_FULL_DENOISE_PASS = 2 }

• enum OptixRayFlags {
    OPTIX_RAY_FLAG_NONE = 0u ,
    OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
    OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u << 1 ,
    OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u << 2 ,
    OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u << 3 ,
    OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u << 4 ,
    OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u << 5 ,
    OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u << 6 ,
    OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u << 7 ,
    OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 10 }

• enum OptixTransformType {
    OPTIX_TRANSFORM_TYPE_NONE = 0 ,
    OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM = 1 ,
    OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM = 2 ,
    OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM = 3 ,
    OPTIX_TRANSFORM_TYPE_INSTANCE = 4 }

• enum OptixTraversableGraphFlags {
    OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0 ,

```

- ```

OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u << 0 ,
OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u << 1 }

```
- enum OptixCompileOptimizationLevel {

```

OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0 ,
OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340 ,
OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341 ,
OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342 ,
OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343 }

```
  - enum OptixCompileDebugLevel {

```

OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT = 0 ,
OPTIX_COMPILE_DEBUG_LEVEL_NONE = 0x2350 ,
OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL = 0x2351 ,
OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353 ,
OPTIX_COMPILE_DEBUG_LEVEL_FULL = 0x2352 }

```
  - enum OptixModuleCompileState {

```

OPTIX_MODULE_COMPILE_STATE_NOT_STARTED = 0x2360 ,
OPTIX_MODULE_COMPILE_STATE_STARTED = 0x2361 ,
OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE = 0x2362 ,
OPTIX_MODULE_COMPILE_STATE_FAILED = 0x2363 ,
OPTIX_MODULE_COMPILE_STATE_COMPLETED = 0x2364 }

```
  - enum OptixPayloadTypeID {

```

OPTIX_PAYLOAD_TYPE_DEFAULT = 0 ,
OPTIX_PAYLOAD_TYPE_ID_0 = (1 << 0u) ,
OPTIX_PAYLOAD_TYPE_ID_1 = (1 << 1u) ,
OPTIX_PAYLOAD_TYPE_ID_2 = (1 << 2u) ,
OPTIX_PAYLOAD_TYPE_ID_3 = (1 << 3u) ,
OPTIX_PAYLOAD_TYPE_ID_4 = (1 << 4u) ,
OPTIX_PAYLOAD_TYPE_ID_5 = (1 << 5u) ,
OPTIX_PAYLOAD_TYPE_ID_6 = (1 << 6u) ,
OPTIX_PAYLOAD_TYPE_ID_7 = (1 << 7u) }

```
  - enum OptixPayloadSemantics {

```

OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE = 0 ,
OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ = 1u << 0 ,
OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE = 2u << 0 ,
OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u << 0 ,
OPTIX_PAYLOAD_SEMANTICS_CH_NONE = 0 ,
OPTIX_PAYLOAD_SEMANTICS_CH_READ = 1u << 2 ,
OPTIX_PAYLOAD_SEMANTICS_CH_WRITE = 2u << 2 ,
OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE = 3u << 2 ,
OPTIX_PAYLOAD_SEMANTICS_MS_NONE = 0 ,
OPTIX_PAYLOAD_SEMANTICS_MS_READ = 1u << 4 ,
OPTIX_PAYLOAD_SEMANTICS_MS_WRITE = 2u << 4 ,
OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE = 3u << 4 ,
OPTIX_PAYLOAD_SEMANTICS_AH_NONE = 0 ,
OPTIX_PAYLOAD_SEMANTICS_AH_READ = 1u << 6 ,
OPTIX_PAYLOAD_SEMANTICS_AH_WRITE = 2u << 6 ,
OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE = 3u << 6 ,
OPTIX_PAYLOAD_SEMANTICS_IS_NONE = 0 ,
OPTIX_PAYLOAD_SEMANTICS_IS_READ = 1u << 8 ,
OPTIX_PAYLOAD_SEMANTICS_IS_WRITE = 2u << 8 ,
OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE = 3u << 8 }

```
  - enum OptixProgramGroupKind {

```

OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421 ,

```

```

OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422 ,
OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423 ,
OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424 ,
OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425 }
• enum OptixProgramGroupFlags { OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0 }
• enum OptixExceptionCodes {
  OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1 ,
  OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED = -3 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE = -5 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT = -6 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT = -7 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE = -8 ,
  OPTIX_EXCEPTION_CODE_INVALID_RAY = -9 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH = -10 ,
  OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH = -11 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT = -12 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD = -13 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD = -14 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS = -15 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0 = -16 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_1 = -17 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2 = -18 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS = -32 ,
  OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH = -33 }
• enum OptixExceptionFlags {
  OPTIX_EXCEPTION_FLAG_NONE = 0 ,
  OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u << 0 ,
  OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u << 1 ,
  OPTIX_EXCEPTION_FLAG_USER = 1u << 2 ,
  OPTIX_EXCEPTION_FLAG_DEBUG = 1u << 3 }
• enum OptixQueryFunctionTableOptions { OPTIX_QUERY_FUNCTION_TABLE_OPTION_
  DUMMY = 0 }

```

### 5.14.1 Detailed Description

OptiX Types.

### 5.14.2 Macro Definition Documentation

#### 5.14.2.1 OPTIX\_AABB\_BUFFER\_BYTE\_ALIGNMENT

```
#define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8u11
```

Alignment requirement for `OptixBuildInputCustomPrimitiveArray::aabbBuffers`.

#### 5.14.2.2 OPTIX\_ACCEL\_BUFFER\_BYTE\_ALIGNMENT

```
#define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128u11
```

Alignment requirement for output and temporary buffers for acceleration structures.

#### 5.14.2.3 OPTIX\_COMPILE\_DEFAULT\_MAX\_PAYLOAD\_TYPE\_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
```



Maximum number of payload types allowed.

#### 5.14.2.4 OPTIX\_COMPILE\_DEFAULT\_MAX\_PAYLOAD\_VALUE\_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
```

Maximum number of payload values allowed.

#### 5.14.2.5 OPTIX\_COMPILE\_DEFAULT\_MAX\_REGISTER\_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
```

Maximum number of registers allowed. Defaults to no explicit limit.

#### 5.14.2.6 OPTIX\_DISPLACEMENT\_MICROMAP\_ARRAY\_BUFFER\_BYTE\_ALIGNMENT

```
#define OPTIX_DISPLACEMENT_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128u11
```

Alignment requirement for displacement micromap array buffers.

#### 5.14.2.7 OPTIX\_DISPLACEMENT\_MICROMAP\_DESC\_BUFFER\_BYTE\_ALIGNMENT

```
#define OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8u11
```

Alignment requirement for displacement micromap descriptor buffers.

#### 5.14.2.8 OPTIX\_DISPLACEMENT\_MICROMAP\_MAX\_SUBDIVISION\_LEVEL

```
#define OPTIX_DISPLACEMENT_MICROMAP_MAX_SUBDIVISION_LEVEL 5
```

Maximum subdivision level for displacement micromaps.

#### 5.14.2.9 OPTIX\_GEOMETRY\_TRANSFORM\_BYTE\_ALIGNMENT

```
#define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16u11
```

Alignment requirement for [OptixBuildInputTriangleArray::preTransform](#).

#### 5.14.2.10 OPTIX\_INSTANCE\_BYTE\_ALIGNMENT

```
#define OPTIX_INSTANCE_BYTE_ALIGNMENT 16u11
```

Alignment requirement for [OptixBuildInputInstanceArray::instances](#).

#### 5.14.2.11 OPTIX\_OPACITY\_MICROMAP\_ARRAY\_BUFFER\_BYTE\_ALIGNMENT

```
#define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128u11
```

Alignment requirement for opacity micromap array buffers.

#### 5.14.2.12 OPTIX\_OPACITY\_MICROMAP\_DESC\_BUFFER\_BYTE\_ALIGNMENT

```
#define OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8u11
```

Alignment requirement for [OptixOpacityMicromapArrayBuildInput::perMicromapDescBuffer](#).

#### 5.14.2.13 OPTIX\_OPACITY\_MICROMAP\_MAX\_SUBDIVISION\_LEVEL

```
#define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
```

Maximum subdivision level for opacity micromaps.

## 5.14.2.14 OPTIX\_OPACITY\_MICROMAP\_PREDEFINED\_INDEX\_FULLY\_OPAQUE

```
#define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE (-2)
```

## 5.14.2.15 OPTIX\_OPACITY\_MICROMAP\_PREDEFINED\_INDEX\_FULLY\_TRANSPARENT

```
#define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT (-1)
```

Predefined index to indicate that a triangle in the BVH build doesn't have an associated opacity micromap, and that it should revert to one of the four possible states for the full triangle.

## 5.14.2.16 OPTIX\_OPACITY\_MICROMAP\_PREDEFINED\_INDEX\_FULLY\_UNKNOWN\_OPAQUE

```
#define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE (-4)
```

## 5.14.2.17 OPTIX\_OPACITY\_MICROMAP\_PREDEFINED\_INDEX\_FULLY\_UNKNOWN\_TRANSPARENT

```
#define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT (-3)
```

## 5.14.2.18 OPTIX\_OPACITY\_MICROMAP\_STATE\_OPAQUE

```
#define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE (1)
```

## 5.14.2.19 OPTIX\_OPACITY\_MICROMAP\_STATE\_TRANSPARENT

```
#define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT (0)
```

Opacity micromaps encode the states of microtriangles in either 1 bit (2-state) or 2 bits (4-state) using the following values.

## 5.14.2.20 OPTIX\_OPACITY\_MICROMAP\_STATE\_UNKNOWN\_OPAQUE

```
#define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE (3)
```

## 5.14.2.21 OPTIX\_OPACITY\_MICROMAP\_STATE\_UNKNOWN\_TRANSPARENT

```
#define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT (2)
```

## 5.14.2.22 OPTIX\_SBT\_RECORD\_ALIGNMENT

```
#define OPTIX_SBT_RECORD_ALIGNMENT 16u11
```

Alignment requirement for device pointers in [OptixShaderBindingTable](#).

## 5.14.2.23 OPTIX\_SBT\_RECORD\_HEADER\_SIZE

```
#define OPTIX_SBT_RECORD_HEADER_SIZE ((size_t)32)
```

Size of the SBT record headers.

## 5.14.2.24 OPTIX\_TRANSFORM\_BYTE\_ALIGNMENT

```
#define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64u11
```

Alignment requirement for [OptixStaticTransform](#), [OptixMatrixMotionTransform](#), [OptixSRTMotionTransform](#).

### 5.14.3 Typedef Documentation

#### 5.14.3.1 CUdeviceptr

```
typedef unsigned long long CUdeviceptr
```

CUDA device pointer.

#### 5.14.3.2 OptixAabb

```
typedef struct OptixAabb OptixAabb
```

AABB inputs.

#### 5.14.3.3 OptixAccelBufferSizes

```
typedef struct OptixAccelBufferSizes OptixAccelBufferSizes
```

Struct for querying builder allocation requirements.

Once queried the sizes should be used to allocate device memory of at least these sizes.

See also [optixAccelComputeMemoryUsage\(\)](#)

#### 5.14.3.4 OptixAccelBuildOptions

```
typedef struct OptixAccelBuildOptions OptixAccelBuildOptions
```

Build options for acceleration structures.

See also [optixAccelComputeMemoryUsage\(\)](#), [optixAccelBuild\(\)](#)

#### 5.14.3.5 OptixAccelEmitDesc

```
typedef struct OptixAccelEmitDesc OptixAccelEmitDesc
```

Specifies a type and output destination for emitted post-build properties.

See also [optixAccelBuild\(\)](#)

#### 5.14.3.6 OptixAccelPropertyType

```
typedef enum OptixAccelPropertyType OptixAccelPropertyType
```

Properties which can be emitted during acceleration structure build.

See also [OptixAccelEmitDesc::type](#).

#### 5.14.3.7 OptixBuildFlags

```
typedef enum OptixBuildFlags OptixBuildFlags
```

Builder Options.

Used for [OptixAccelBuildOptions::buildFlags](#). Can be or'ed together.

#### 5.14.3.8 OptixBuildInput

```
typedef struct OptixBuildInput OptixBuildInput
```

Build inputs.

All of them support motion and the size of the data arrays needs to match the number of motion steps

See also [optixAccelComputeMemoryUsage\(\)](#), [optixAccelBuild\(\)](#)

### 5.14.3.9 OptixBuildInputCurveArray

```
typedef struct OptixBuildInputCurveArray OptixBuildInputCurveArray
```

Curve inputs.

A curve is a swept surface defined by a 3D spline curve and a varying width (radius). A curve (or "strand") of degree  $d$  (3=cubic, 2=quadratic, 1=linear) is represented by  $N > d$  vertices and  $N$  width values, and comprises  $N - d$  segments. Each segment is defined by  $d+1$  consecutive vertices. Each curve may have a different number of vertices.

OptiX describes the curve array as a list of curve segments. The primitive id is the segment number. It is the user's responsibility to maintain a mapping between curves and curve segments. Each index buffer entry  $i = \text{indexBuffer}[\text{primid}]$  specifies the start of a curve segment, represented by  $d+1$  consecutive vertices in the vertex buffer, and  $d+1$  consecutive widths in the width buffer. Width is interpolated the same way vertices are interpolated, that is, using the curve basis.

Each curves build input has only one SBT record. To create curves with different materials in the same BVH, use multiple build inputs.

See also [OptixBuildInput::curveArray](#)

### 5.14.3.10 OptixBuildInputCustomPrimitiveArray

```
typedef struct OptixBuildInputCustomPrimitiveArray
OptixBuildInputCustomPrimitiveArray
```

Custom primitive inputs.

See also [OptixBuildInput::customPrimitiveArray](#)

### 5.14.3.11 OptixBuildInputDisplacementMicromap

```
typedef struct OptixBuildInputDisplacementMicromap
OptixBuildInputDisplacementMicromap
```

Optional displacement part of a triangle array input.

### 5.14.3.12 OptixBuildInputInstanceArray

```
typedef struct OptixBuildInputInstanceArray OptixBuildInputInstanceArray
```

Instance and instance pointer inputs.

See also [OptixBuildInput::instanceArray](#)

### 5.14.3.13 OptixBuildInputOpacityMicromap

```
typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap
```

### 5.14.3.14 OptixBuildInputSphereArray

```
typedef struct OptixBuildInputSphereArray OptixBuildInputSphereArray
```

Sphere inputs.

A sphere is defined by a center point and a radius. Each center point is represented by a vertex in the vertex buffer. There is either a single radius for all spheres, or the radii are represented by entries in the radius buffer.

The vertex buffers and radius buffers point to a host array of device pointers, one per motion step. Host array size must match the number of motion keys as set in [OptixMotionOptions](#) (or an array of size 1 if

`OptixMotionOptions::numKeys` is set to 0 or 1). Each per motion key device pointer must point to an array of vertices corresponding to the center points of the spheres, or an array of 1 or N radii. Format `OPTIX_VERTEX_FORMAT_FLOAT3` is used for vertices, `OPTIX_VERTEX_FORMAT_FLOAT` for radii.

See also `OptixBuildInput::sphereArray`

#### 5.14.3.15 `OptixBuildInputTriangleArray`

```
typedef struct OptixBuildInputTriangleArray OptixBuildInputTriangleArray
```

Triangle inputs.

See also `OptixBuildInput::triangleArray`

#### 5.14.3.16 `OptixBuildInputType`

```
typedef enum OptixBuildInputType OptixBuildInputType
```

Enum to distinguish the different build input types.

See also `OptixBuildInput::type`

#### 5.14.3.17 `OptixBuildOperation`

```
typedef enum OptixBuildOperation OptixBuildOperation
```

Enum to specify the acceleration build operation.

Used in `OptixAccelBuildOptions`, which is then passed to `optixAccelBuild` and `optixAccelComputeMemoryUsage`, this enum indicates whether to do a build or an update of the acceleration structure.

Acceleration structure updates utilize the same acceleration structure, but with updated bounds. Updates are typically much faster than builds, however, large perturbations can degrade the quality of the acceleration structure.

See also `optixAccelComputeMemoryUsage()`, `optixAccelBuild()`, `OptixAccelBuildOptions`

#### 5.14.3.18 `OptixBuiltinISOptions`

```
typedef struct OptixBuiltinISOptions OptixBuiltinISOptions
```

Specifies the options for retrieving an intersection program for a built-in primitive type. The primitive type must not be `OPTIX_PRIMITIVE_TYPE_CUSTOM`.

See also `optixBuiltinISModuleGet()`

#### 5.14.3.19 `OptixCompileDebugLevel`

```
typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
```

Debug levels.

See also `OptixModuleCompileOptions::debugLevel`

#### 5.14.3.20 `OptixCompileOptimizationLevel`

```
typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
```

Optimization levels.

See also `OptixModuleCompileOptions::optLevel`

#### 5.14.3.21 OptixCurveEndcapFlags

```
typedef enum OptixCurveEndcapFlags OptixCurveEndcapFlags
```

Curve end cap types, for non-linear curves.

#### 5.14.3.22 OptixDenoiser

```
typedef struct OptixDenoiser_t* OptixDenoiser
```

Opaque type representing a denoiser instance.

#### 5.14.3.23 OptixDenoiserAlphaMode

```
typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
```

Various parameters used by the denoiser.

See also [optixDenoiserInvoke\(\)](#)

[optixDenoiserComputeIntensity\(\)](#)

[optixDenoiserComputeAverageColor\(\)](#)

#### 5.14.3.24 OptixDenoiserAOVType

```
typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
```

AOV type used by the denoiser.

#### 5.14.3.25 OptixDenoiserGuideLayer

```
typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
```

Guide layer for the denoiser.

See also [optixDenoiserInvoke\(\)](#)

#### 5.14.3.26 OptixDenoiserLayer

```
typedef struct OptixDenoiserLayer OptixDenoiserLayer
```

Input/Output layers for the denoiser.

See also [optixDenoiserInvoke\(\)](#)

#### 5.14.3.27 OptixDenoiserModelKind

```
typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
```

Model kind used by the denoiser.

See also [optixDenoiserCreate](#)

#### 5.14.3.28 OptixDenoiserOptions

```
typedef struct OptixDenoiserOptions OptixDenoiserOptions
```

Options used by the denoiser.

See also [optixDenoiserCreate\(\)](#)

#### 5.14.3.29 OptixDenoiserParams

```
typedef struct OptixDenoiserParams OptixDenoiserParams
```

#### 5.14.3.30 OptixDenoiserSizes

```
typedef struct OptixDenoiserSizes OptixDenoiserSizes
```

Various sizes related to the denoiser.

See also [optixDenoiserComputeMemoryResources\(\)](#)

#### 5.14.3.31 OptixDeviceContext

```
typedef struct OptixDeviceContext_t* OptixDeviceContext
```

Opaque type representing a device context.

#### 5.14.3.32 OptixDeviceContextOptions

```
typedef struct OptixDeviceContextOptions OptixDeviceContextOptions
```

Parameters used for [optixDeviceContextCreate\(\)](#)

See also [optixDeviceContextCreate\(\)](#)

#### 5.14.3.33 OptixDeviceContextValidationMode

```
typedef enum OptixDeviceContextValidationMode  
OptixDeviceContextValidationMode
```

Validation mode settings.

When enabled, certain device code utilities will be enabled to provide as good debug and error checking facilities as possible.

See also [optixDeviceContextCreate\(\)](#)

#### 5.14.3.34 OptixDeviceProperty

```
typedef enum OptixDeviceProperty OptixDeviceProperty
```

Parameters used for [optixDeviceContextGetProperty\(\)](#)

See also [optixDeviceContextGetProperty\(\)](#)

#### 5.14.3.35 OptixDisplacementMicromapArrayBuildInput

```
typedef struct OptixDisplacementMicromapArrayBuildInput  
OptixDisplacementMicromapArrayBuildInput
```

Inputs to displacement micromaps array construction.

#### 5.14.3.36 OptixDisplacementMicromapArrayIndexingMode

```
typedef enum OptixDisplacementMicromapArrayIndexingMode  
OptixDisplacementMicromapArrayIndexingMode
```

indexing mode of triangles to displacement micromaps in an array, used in [OptixBuildInputDisplacementMicromap](#).

#### 5.14.3.37 OptixDisplacementMicromapBiasAndScaleFormat

```
typedef enum OptixDisplacementMicromapBiasAndScaleFormat
OptixDisplacementMicromapBiasAndScaleFormat
```

#### 5.14.3.38 OptixDisplacementMicromapDesc

```
typedef struct OptixDisplacementMicromapDesc OptixDisplacementMicromapDesc
```

#### 5.14.3.39 OptixDisplacementMicromapDirectionFormat

```
typedef enum OptixDisplacementMicromapDirectionFormat
OptixDisplacementMicromapDirectionFormat
```

#### 5.14.3.40 OptixDisplacementMicromapFlags

```
typedef enum OptixDisplacementMicromapFlags OptixDisplacementMicromapFlags
```

Flags defining behavior of DMMs in a DMM array.

#### 5.14.3.41 OptixDisplacementMicromapFormat

```
typedef enum OptixDisplacementMicromapFormat OptixDisplacementMicromapFormat
```

DMM input data format.

#### 5.14.3.42 OptixDisplacementMicromapHistogramEntry

```
typedef struct OptixDisplacementMicromapHistogramEntry
OptixDisplacementMicromapHistogramEntry
```

Displacement micromap histogram entry. Specifies how many displacement micromaps of a specific type are input to the displacement micromap array build. Note that while this is similar to [OptixDisplacementMicromapUsageCount](#), the histogram entry specifies how many displacement micromaps of a specific type are combined into a displacement micromap array.

#### 5.14.3.43 OptixDisplacementMicromapTriangleFlags

```
typedef enum OptixDisplacementMicromapTriangleFlags
OptixDisplacementMicromapTriangleFlags
```

#### 5.14.3.44 OptixDisplacementMicromapUsageCount

```
typedef struct OptixDisplacementMicromapUsageCount
OptixDisplacementMicromapUsageCount
```

Displacement micromap usage count for acceleration structure builds. Specifies how many displacement micromaps of a specific type are referenced by triangles when building the AS. Note that while this is similar to [OptixDisplacementMicromapHistogramEntry](#), the usage count specifies how many displacement micromaps of a specific type are referenced by triangles in the AS.

#### 5.14.3.45 OptixExceptionCodes

```
typedef enum OptixExceptionCodes OptixExceptionCodes
```

The following values are used to indicate which exception was thrown.



#### 5.14.3.46 OptixExceptionFlags

```
typedef enum OptixExceptionFlags OptixExceptionFlags
```

Exception flags.

See also `OptixPipelineCompileOptions::exceptionFlags`, `OptixExceptionCodes`

#### 5.14.3.47 OptixGeometryFlags

```
typedef enum OptixGeometryFlags OptixGeometryFlags
```

Flags used by `OptixBuildInputTriangleArray::flags` and `OptixBuildInputCustomPrimitiveArray::flags`.

#### 5.14.3.48 OptixHitKind

```
typedef enum OptixHitKind OptixHitKind
```

Legacy type: A subset of the hit kinds for built-in primitive intersections. It is preferred to use `optixGetPrimitiveType()`, together with `optixIsFrontFaceHit()` or `optixIsBackFaceHit()`.

See also `optixGetHitKind()`

#### 5.14.3.49 OptixImage2D

```
typedef struct OptixImage2D OptixImage2D
```

Image descriptor used by the denoiser.

See also `optixDenoiserInvoke()`, `optixDenoiserComputeIntensity()`

#### 5.14.3.50 OptixIndicesFormat

```
typedef enum OptixIndicesFormat OptixIndicesFormat
```

Format of indices used in `OptixBuildInputTriangleArray::indexFormat`.

#### 5.14.3.51 OptixInstance

```
typedef struct OptixInstance OptixInstance
```

Instances.

See also `OptixBuildInputInstanceArray::instances`

#### 5.14.3.52 OptixInstanceFlags

```
typedef enum OptixInstanceFlags OptixInstanceFlags
```

Flags set on the `OptixInstance::flags`.

These can be or'ed together to combine multiple flags.

#### 5.14.3.53 OptixLogCallback

```
typedef void(* OptixLogCallback) (unsigned int level, const char *tag, const char *message, void *cbdata)
```

Type of the callback function used for log messages.

Parameters

in	<i>level</i>	The log level indicates the severity of the message. See below for possible values.
----	--------------	---

## Parameters

in	<i>tag</i>	A terse message category description (e.g., 'SCENE STAT').
in	<i>message</i>	Null terminated log message (without newline at the end).
in	<i>cbdata</i>	Callback data that was provided with the callback pointer.

It is the users responsibility to ensure thread safety within this function.

The following log levels are defined.

0 disable Setting the callback level will disable all messages. The callback function will not be called in this case. 1 fatal A non-recoverable error. The context and/or OptiX itself might no longer be in a usable state. 2 error A recoverable error, e.g., when passing invalid call parameters. 3 warning Hints that OptiX might not behave exactly as requested by the user or may perform slower than expected. 4 print Status or progress messages.

Higher levels might occur.

See also [optixDeviceContextSetLogCallback\(\)](#), [OptixDeviceContextOptions](#)

#### 5.14.3.54 OptixMatrixMotionTransform

```
typedef struct OptixMatrixMotionTransform OptixMatrixMotionTransform
```

Represents a matrix motion transformation.

The device address of instances of this type must be a multiple of OPTIX\_TRANSFORM\_BYTE\_ALIGNMENT.

This struct, as defined here, handles only N=2 motion keys due to the fixed array length of its transform member. The following example shows how to create instances for an arbitrary number N of motion keys:

```
float matrixData[N][12];
... // setup matrixData
size_t transformSizeInBytes = sizeof(OptixMatrixMotionTransform) + (N-2) * 12 * sizeof(float);
OptixMatrixMotionTransform* matrixMoptionTransform = (OptixMatrixMotionTransform*)
malloc(transformSizeInBytes);
memset(matrixMoptionTransform, 0, transformSizeInBytes);
... // setup other members of matrixMoptionTransform
matrixMoptionTransform->motionOptions.numKeys
memcpy(matrixMoptionTransform->transform, matrixData, N * 12 * sizeof(float));
... // copy matrixMoptionTransform to device memory
free(matrixMoptionTransform)
```

See also [optixConvertPointerToTraversableHandle\(\)](#)

#### 5.14.3.55 OptixMicromapBuffers

```
typedef struct OptixMicromapBuffers OptixMicromapBuffers
```

Buffer inputs for opacity/displacement micromap array builds.

#### 5.14.3.56 OptixMicromapBufferSizes

```
typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
```

Conservative memory requirements for building a opacity/displacement micromap array.

#### 5.14.3.57 OptixModule

```
typedef struct OptixModule_t* OptixModule
```

Opaque type representing a module.

#### 5.14.3.58 OptixModuleCompileBoundValueEntry

```
typedef struct OptixModuleCompileBoundValueEntry
OptixModuleCompileBoundValueEntry
```

Struct for specifying specializations for pipelineParams as specified in [OptixPipelineCompileOptions::pipelineLaunchParamsVariableName](#).

The bound values are supposed to represent a constant value in the pipelineParams. OptiX will attempt to locate all loads from the pipelineParams and correlate them to the appropriate bound value, but there are cases where OptiX cannot safely or reliably do this. For example if the pointer to the pipelineParams is passed as an argument to a non-inline function or the offset of the load to the pipelineParams cannot be statically determined (e.g. accessed in a loop). No module should rely on the value being specialized in order to work correctly. The values in the pipelineParams specified on optixLaunch should match the bound value. If validation mode is enabled on the context, OptiX will verify that the bound values specified matches the values in pipelineParams specified to optixLaunch.

These values are compiled in to the module as constants. Once the constants are inserted into the code, an optimization pass will be run that will attempt to propagate the constants and remove unreachable code.

If caching is enabled, changes in these values will result in newly compiled modules.

The pipelineParamOffset and sizeInBytes must be within the bounds of the pipelineParams variable. OPTIX\_ERROR\_INVALID\_VALUE will be returned from optixModuleCreate otherwise.

If more than one bound value overlaps or the size of a bound value is equal to 0, an OPTIX\_ERROR\_INVALID\_VALUE will be returned from optixModuleCreate.

The same set of bound values do not need to be used for all modules in a pipeline, but overlapping values between modules must have the same value. OPTIX\_ERROR\_INVALID\_VALUE will be returned from optixPipelineCreate otherwise.

See also [OptixModuleCompileOptions](#)

#### 5.14.3.59 OptixModuleCompileOptions

```
typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
```

Compilation options for module.

See also [optixModuleCreate\(\)](#)

#### 5.14.3.60 OptixModuleCompileState

```
typedef enum OptixModuleCompileState OptixModuleCompileState
```

Module compilation state.

See also [optixModuleGetCompilationState\(\)](#), [optixModuleCreateWithTasks\(\)](#)

#### 5.14.3.61 OptixMotionFlags

```
typedef enum OptixMotionFlags OptixMotionFlags
```

Enum to specify motion flags.

See also [OptixMotionOptions::flags](#).

#### 5.14.3.62 OptixMotionOptions

```
typedef struct OptixMotionOptions OptixMotionOptions
```

Motion options.

See also [OptixAccelBuildOptions::motionOptions](#), [OptixMatrixMotionTransform::motionOptions](#), [OptixSRTMotionTransform::motionOptions](#)

#### 5.14.3.63 OptixOpacityMicromapArrayBuildInput

```
typedef struct OptixOpacityMicromapArrayBuildInput
OptixOpacityMicromapArrayBuildInput
```

Inputs to opacity micromap array construction.

#### 5.14.3.64 OptixOpacityMicromapArrayIndexingMode

```
typedef enum OptixOpacityMicromapArrayIndexingMode
OptixOpacityMicromapArrayIndexingMode
```

indexing mode of triangles to opacity micromaps in an array, used in [OptixBuildInputOpacityMicromap](#).

#### 5.14.3.65 OptixOpacityMicromapDesc

```
typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc
```

Opacity micromap descriptor.

#### 5.14.3.66 OptixOpacityMicromapFlags

```
typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags
```

Flags defining behavior of opacity micromaps in a opacity micromap array.

#### 5.14.3.67 OptixOpacityMicromapFormat

```
typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat
```

Specifies whether to use a 2- or 4-state opacity micromap format.

#### 5.14.3.68 OptixOpacityMicromapHistogramEntry

```
typedef struct OptixOpacityMicromapHistogramEntry
OptixOpacityMicromapHistogramEntry
```

Opacity micromap histogram entry. Specifies how many opacity micromaps of a specific type are input to the opacity micromap array build. Note that while this is similar to [OptixOpacityMicromapUsageCount](#), the histogram entry specifies how many opacity micromaps of a specific type are combined into a opacity micromap array.

#### 5.14.3.69 OptixOpacityMicromapUsageCount

```
typedef struct OptixOpacityMicromapUsageCount OptixOpacityMicromapUsageCount
```

Opacity micromap usage count for acceleration structure builds. Specifies how many opacity micromaps of a specific type are referenced by triangles when building the AS. Note that while this is similar to [OptixOpacityMicromapHistogramEntry](#), the usage count specifies how many opacity micromaps of a specific type are referenced by triangles in the AS.

#### 5.14.3.70 OptixPayloadSemantics

```
typedef enum OptixPayloadSemantics OptixPayloadSemantics
```

Semantic flags for a single payload word.

Used to specify the semantics of a payload word per shader type. "read": Shader of this type may read the payload word. "write": Shader of this type may write the payload word.

"trace\_caller\_write": Shaders may consume the value of the payload word passed to `optixTrace` by the caller. "trace\_caller\_read": The caller to `optixTrace` may read the payload word after the call to `optixTrace`.

Semantics can be bitwise combined. Combining "read" and "write" is equivalent to specifying "read\_write". A payload needs to be writable by the caller or at least one shader type. A payload needs to be readable by the caller or at least one shader type after a being writable.

#### 5.14.3.71 OptixPayloadType

```
typedef struct OptixPayloadType OptixPayloadType
```

Specifies a single payload type.

#### 5.14.3.72 OptixPayloadTypeID

```
typedef enum OptixPayloadTypeID OptixPayloadTypeID
```

Payload type identifiers.

#### 5.14.3.73 OptixPipeline

```
typedef struct OptixPipeline_t* OptixPipeline
```

Opaque type representing a pipeline.

#### 5.14.3.74 OptixPipelineCompileOptions

```
typedef struct OptixPipelineCompileOptions OptixPipelineCompileOptions
```

Compilation options for all modules of a pipeline.

Similar to `OptixModuleCompileOptions`, but these options here need to be equal for all modules of a pipeline.

See also `optixModuleCreate()`, `optixPipelineCreate()`

#### 5.14.3.75 OptixPipelineLinkOptions

```
typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
```

Link options for a pipeline.

See also `optixPipelineCreate()`

#### 5.14.3.76 OptixPixelFormat

```
typedef enum OptixPixelFormat OptixPixelFormat
```

Pixel formats used by the denoiser.

See also `OptixImage2D::format`

#### 5.14.3.77 OptixPrimitiveType

```
typedef enum OptixPrimitiveType OptixPrimitiveType
```

Builtin primitive types.

#### 5.14.3.78 OptixPrimitiveTypeFlags

```
typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags
```

Builtin flags may be bitwise combined.

See also [OptixPipelineCompileOptions::usesPrimitiveTypeFlags](#)

#### 5.14.3.79 OptixProgramGroup

```
typedef struct OptixProgramGroup_t* OptixProgramGroup
```

Opaque type representing a program group.

#### 5.14.3.80 OptixProgramGroupCallables

```
typedef struct OptixProgramGroupCallables OptixProgramGroupCallables
```

Program group representing callables.

Module and entry function name need to be valid for at least one of the two callables.

See also [#OptixProgramGroupDesc::callables](#)

#### 5.14.3.81 OptixProgramGroupDesc

```
typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
```

Descriptor for program groups.

#### 5.14.3.82 OptixProgramGroupFlags

```
typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
```

Flags for program groups.

#### 5.14.3.83 OptixProgramGroupHitgroup

```
typedef struct OptixProgramGroupHitgroup OptixProgramGroupHitgroup
```

Program group representing the hitgroup.

For each of the three program types, module and entry function name might both be nullptr.

See also [OptixProgramGroupDesc::hitgroup](#)

#### 5.14.3.84 OptixProgramGroupKind

```
typedef enum OptixProgramGroupKind OptixProgramGroupKind
```

Distinguishes different kinds of program groups.

#### 5.14.3.85 OptixProgramGroupOptions

```
typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
```

Program group options.

See also `optixProgramGroupCreate()`

#### 5.14.3.86 OptixProgramGroupSingleModule

```
typedef struct OptixProgramGroupSingleModule OptixProgramGroupSingleModule
```

Program group representing a single module.

Used for raygen, miss, and exception programs. In case of raygen and exception programs, module and entry function name need to be valid. For miss programs, module and entry function name might both be `nullptr`.

See also `OptixProgramGroupDesc::raygen`, `OptixProgramGroupDesc::miss`, `OptixProgramGroupDesc::exception`

#### 5.14.3.87 OptixQueryFunctionTable\_t

```
typedef OptixResult() OptixQueryFunctionTable_t(int abiId, unsigned int
numOptions, OptixQueryFunctionTableOptions *, const void **, void
*functionTable, size_t sizeOfTable)
```

Type of the function `optixQueryFunctionTable()`

#### 5.14.3.88 OptixQueryFunctionTableOptions

```
typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
```

Options that can be passed to `optixQueryFunctionTable()`

#### 5.14.3.89 OptixRayFlags

```
typedef enum OptixRayFlags OptixRayFlags
```

Ray flags passed to the device function `optixTrace()`. These affect the behavior of traversal per invocation.

See also `optixTrace()`

#### 5.14.3.90 OptixRelocateInput

```
typedef struct OptixRelocateInput OptixRelocateInput
```

Relocation inputs.

See also `optixAccelRelocate()`

#### 5.14.3.91 OptixRelocateInputInstanceArray

```
typedef struct OptixRelocateInputInstanceArray
OptixRelocateInputInstanceArray
```

Instance and instance pointer inputs.

See also `OptixRelocateInput::instanceArray`

#### 5.14.3.92 OptixRelocateInputOpacityMicromap

```
typedef struct OptixRelocateInputOpacityMicromap
OptixRelocateInputOpacityMicromap
```

### 5.14.3.93 OptixRelocateInputTriangleArray

```
typedef struct OptixRelocateInputTriangleArray
OptixRelocateInputTriangleArray
```

Triangle inputs.

See also [OptixRelocateInput::triangleArray](#)

### 5.14.3.94 OptixRelocationInfo

```
typedef struct OptixRelocationInfo OptixRelocationInfo
```

Used to store information related to relocation of optix data structures.

See also [optixOpacityMicromapArrayGetRelocationInfo\(\)](#), [optixOpacityMicromapArrayRelocate\(\)](#), [optixAccelGetRelocationInfo\(\)](#), [optixAccelRelocate\(\)](#), [optixCheckRelocationCompatibility\(\)](#)

### 5.14.3.95 OptixResult

```
typedef enum OptixResult OptixResult
```

Result codes returned from API functions.

All host side API functions return `OptixResult` with the exception of [optixGetErrorName](#) and [optixGetErrorString](#). When successful `OPTIX_SUCCESS` is returned. All return codes except for `OPTIX_SUCCESS` should be assumed to be errors as opposed to a warning.

See also [optixGetErrorName\(\)](#), [optixGetErrorString\(\)](#)

### 5.14.3.96 OptixShaderBindingTable

```
typedef struct OptixShaderBindingTable OptixShaderBindingTable
```

Describes the shader binding table (SBT)

See also [optixLaunch\(\)](#)

### 5.14.3.97 OptixSRTData

```
typedef struct OptixSRTData OptixSRTData
```

Represents an SRT transformation.

An SRT transformation can represent a smooth rotation with fewer motion keys than a matrix transformation. Each motion key is constructed from elements taken from a matrix *S*, a quaternion *R*, and a translation *T*.

The scaling matrix  $S = \begin{bmatrix} sx & a & b & pvx \\ 0 & sy & c & pvy \\ 0 & 0 & sz & pvz \end{bmatrix}$  defines an affine transformation that can include scale, shear, and a translation. The translation allows to define the pivot point for the subsequent rotation.

The quaternion  $R = [qx, qy, qz, qw]$  describes a rotation with angular component  $qw = \cos(\theta/2)$  and other components  $[qx, qy, qz] = \sin(\theta/2) * [ax, ay, az]$  where the axis  $[ax, ay, az]$  is normalized.

The translation matrix  $T = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \end{bmatrix}$  defines another translation that is applied after the rotation.

Typically, this translation includes the inverse translation from the matrix *S* to reverse the translation



for the pivot point for R.

To obtain the effective transformation at time  $t$ , the elements of the components of S, R, and T will be interpolated linearly. The components are then multiplied to obtain the combined transformation  $C = T * R * S$ . The transformation C is the effective object-to-world transformations at time  $t$ , and  $C^{-1}$  is the effective world-to-object transformation at time  $t$ .

See also [OptixSRTMotionTransform::srtData](#), [optixConvertPointerToTraversableHandle\(\)](#)

#### 5.14.3.98 OptixSRTMotionTransform

```
typedef struct OptixSRTMotionTransform OptixSRTMotionTransform
```

Represents an SRT motion transformation.

The device address of instances of this type must be a multiple of OPTIX\_TRANSFORM\_BYTE\_ALIGNMENT.

This struct, as defined here, handles only N=2 motion keys due to the fixed array length of its srtData member. The following example shows how to create instances for an arbitrary number N of motion keys:

```
OptixSRTData srtData[N];
... // setup srtData
size_t transformSizeInBytes = sizeof(OptixSRTMotionTransform) + (N-2) * sizeof(OptixSRTData);
OptixSRTMotionTransform* srtMotionTransform = (OptixSRTMotionTransform*) malloc(transformSizeInBytes);
memset(srtMotionTransform, 0, transformSizeInBytes);
... // setup other members of srtMotionTransform
srtMotionTransform->motionOptions.numKeys = N;
memcpy(srtMotionTransform->srtData, srtData, N * sizeof(OptixSRTData));
... // copy srtMotionTransform to device memory
free(srtMotionTransform)
```

See also [optixConvertPointerToTraversableHandle\(\)](#)

#### 5.14.3.99 OptixStackSizes

```
typedef struct OptixStackSizes OptixStackSizes
```

Describes the stack size requirements of a program group.

See also [optixProgramGroupGetStackSize\(\)](#)

#### 5.14.3.100 OptixStaticTransform

```
typedef struct OptixStaticTransform OptixStaticTransform
```

Static transform.

The device address of instances of this type must be a multiple of OPTIX\_TRANSFORM\_BYTE\_ALIGNMENT.

See also [optixConvertPointerToTraversableHandle\(\)](#)

#### 5.14.3.101 OptixTask

```
typedef struct OptixTask_t* OptixTask
```

Opaque type representing a work task.

#### 5.14.3.102 OptixTransformFormat

```
typedef enum OptixTransformFormat OptixTransformFormat
```

Format of transform used in [OptixBuildInputTriangleArray::transformFormat](#).

### 5.14.3.103 OptixTransformType

```
typedef enum OptixTransformType OptixTransformType
```

Transform.

OptixTransformType is used by the device function `optixGetTransformTypeFromHandle()` to determine the type of the `OptixTraversableHandle` returned from `optixGetTransformListHandle()`.

### 5.14.3.104 OptixTraversableGraphFlags

```
typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags
```

Specifies the set of valid traversable graphs that may be passed to invocation of `optixTrace()`. Flags may be bitwise combined.

### 5.14.3.105 OptixTraversableHandle

```
typedef unsigned long long OptixTraversableHandle
```

Traversable handle.

### 5.14.3.106 OptixTraversableType

```
typedef enum OptixTraversableType OptixTraversableType
```

Traversable Handles.

See also `optixConvertPointerToTraversableHandle()`

### 5.14.3.107 OptixVertexFormat

```
typedef enum OptixVertexFormat OptixVertexFormat
```

Format of vertices used in `OptixBuildInputTriangleArray::vertexFormat`.

### 5.14.3.108 OptixVisibilityMask

```
typedef unsigned int OptixVisibilityMask
```

Visibility mask.

## 5.14.4 Enumeration Type Documentation

### 5.14.4.1 OptixAccelPropertyType

```
enum OptixAccelPropertyType
```

Properties which can be emitted during acceleration structure build.

See also `OptixAccelEmitDesc::type`.

Enumerator

OPTIX_PROPERTY_TYPE_COMPACTED_SIZE	Size of a compacted acceleration structure. The device pointer points to a uint64.
OPTIX_PROPERTY_TYPE_AABBS	<code>OptixAabb * numMotionSteps</code> .

### 5.14.4.2 OptixBuildFlags

enum `OptixBuildFlags`

Builder Options.

Used for `OptixAccelBuildOptions::buildFlags`. Can be or'ed together.

Enumerator

<code>OPTIX_BUILD_FLAG_NONE</code>	No special flags set.
<code>OPTIX_BUILD_FLAG_ALLOW_UPDATE</code>	Allow updating the build with new vertex positions with subsequent calls to <code>optixAccelBuild</code> .
<code>OPTIX_BUILD_FLAG_ALLOW_COMPACTION</code>	
<code>OPTIX_BUILD_FLAG_PREFER_FAST_TRACE</code>	This flag is mutually exclusive with <code>OPTIX_BUILD_FLAG_PREFER_FAST_BUILD</code> .
<code>OPTIX_BUILD_FLAG_PREFER_FAST_BUILD</code>	This flag is mutually exclusive with <code>OPTIX_BUILD_FLAG_PREFER_FAST_TRACE</code> .
<code>OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS</code>	Allow random access to build input vertices See <code>optixGetTriangleVertexData</code> <code>optixGetLinearCurveVertexData</code> <code>optixGetQuadraticBSplineVertexData</code> <code>optixGetCubicBSplineVertexData</code> <code>optixGetCatmullRomVertexData</code> <code>optixGetRibbonVertexData</code> <code>optixGetRibbonNormal</code> <code>optixGetSphereData</code> .
<code>OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS</code>	Allow random access to instances See <code>optixGetInstanceTraversableFromIAS</code> .
<code>OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE</code>	Support updating the opacity micromap array and opacity micromap indices on refits. May increase AS size and may have a small negative impact on traversal performance. If this flag is absent, all opacity micromap inputs must remain unchanged between the initial AS builds and their subsequent refits.
<code>OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS</code>	If enabled, any instances referencing this GAS are allowed to disable the opacity micromap test through the <code>DISABLE_OPACITY_MICROMAPS</code> flag instance flag. Note that the GAS will not be optimized for the attached opacity micromap Arrays if this flag is set, which may result in reduced traversal performance.

### 5.14.4.3 OptixBuildInputType

enum `OptixBuildInputType`

Enum to distinguish the different build input types.

See also `OptixBuildInput::type`

## Enumerator

OPTIX_BUILD_INPUT_TYPE_TRIANGLES	Triangle inputs. See also <a href="#">OptixBuildInputTriangleArray</a>
OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES	Custom primitive inputs. See also <a href="#">OptixBuildInputCustomPrimitiveArray</a>
OPTIX_BUILD_INPUT_TYPE_INSTANCES	Instance inputs. See also <a href="#">OptixBuildInputInstanceArray</a>
OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS	Instance pointer inputs. See also <a href="#">OptixBuildInputInstanceArray</a>
OPTIX_BUILD_INPUT_TYPE_CURVES	Curve inputs. See also <a href="#">OptixBuildInputCurveArray</a>
OPTIX_BUILD_INPUT_TYPE_SPHERES	Sphere inputs. See also <a href="#">OptixBuildInputSphereArray</a>

## 5.14.4.4 OptixBuildOperation

enum [OptixBuildOperation](#)

Enum to specify the acceleration build operation.

Used in [OptixAccelBuildOptions](#), which is then passed to `optixAccelBuild` and `optixAccelComputeMemoryUsage`, this enum indicates whether to do a build or an update of the acceleration structure.

Acceleration structure updates utilize the same acceleration structure, but with updated bounds. Updates are typically much faster than builds, however, large perturbations can degrade the quality of the acceleration structure.

See also [optixAccelComputeMemoryUsage\(\)](#), [optixAccelBuild\(\)](#), [OptixAccelBuildOptions](#)

## Enumerator

OPTIX_BUILD_OPERATION_BUILD	Perform a full build operation.
OPTIX_BUILD_OPERATION_UPDATE	Perform an update using new bounds.

## 5.14.4.5 OptixCompileDebugLevel

enum [OptixCompileDebugLevel](#)

Debug levels.

See also [OptixModuleCompileOptions::debugLevel](#)

## Enumerator

OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT	Default currently is minimal.
OPTIX_COMPILE_DEBUG_LEVEL_NONE	No debug information.
OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL	Generate information that does not impact performance. Note this replaces OPTIX_COMPILE_DEBUG_LEVEL_LINEINFO.

## Enumerator

OPTIX_COMPILE_DEBUG_LEVEL_MODERATE	Generate some debug information with slight performance cost.
OPTIX_COMPILE_DEBUG_LEVEL_FULL	Generate full debug information.

## 5.14.4.6 OptixCompileOptimizationLevel

enum [OptixCompileOptimizationLevel](#)

Optimization levels.

See also [OptixModuleCompileOptions::optLevel](#)

## Enumerator

OPTIX_COMPILE_OPTIMIZATION_DEFAULT	Default is to run all optimizations.
OPTIX_COMPILE_OPTIMIZATION_LEVEL_0	No optimizations.
OPTIX_COMPILE_OPTIMIZATION_LEVEL_1	Some optimizations.
OPTIX_COMPILE_OPTIMIZATION_LEVEL_2	Most optimizations.
OPTIX_COMPILE_OPTIMIZATION_LEVEL_3	All optimizations.

## 5.14.4.7 OptixCurveEndcapFlags

enum [OptixCurveEndcapFlags](#)

Curve end cap types, for non-linear curves.

## Enumerator

OPTIX_CURVE_ENDCAP_DEFAULT	Default end caps. Round end caps for linear, no end caps for quadratic/cubic.
OPTIX_CURVE_ENDCAP_ON	Flat end caps at both ends of quadratic/cubic curve segments. Not valid for linear.

## 5.14.4.8 OptixDenoiserAlphaMode

enum [OptixDenoiserAlphaMode](#)

Various parameters used by the denoiser.

See also [optixDenoiserInvoke\(\)](#)

[optixDenoiserComputeIntensity\(\)](#)

[optixDenoiserComputeAverageColor\(\)](#)

## Enumerator

OPTIX_DENOISER_ALPHA_MODE_COPY	Copy alpha (if present) from input layer, no denoising.
OPTIX_DENOISER_ALPHA_MODE_ALPHA_AS_AOV	Denoise alpha separately. With AOV model kinds, treat alpha like an AOV.

## Enumerator

OPTIX_DENOISER_ALPHA_MODE_FULL_DENOISE_PASS	With AOV model kinds, full denoise pass with alpha. This is slower than OPTIX_DENOISER_ALPHA_MODE_ALPHA_AS_AOV.
---	---

## 5.14.4.9 OptixDenoiserAOVType

enum [OptixDenoiserAOVType](#)

AOV type used by the denoiser.

## Enumerator

OPTIX_DENOISER_AOV_TYPE_NONE	Unspecified AOV type.
OPTIX_DENOISER_AOV_TYPE_BEAUTY	
OPTIX_DENOISER_AOV_TYPE_SPECULAR	
OPTIX_DENOISER_AOV_TYPE_REFLECTION	
OPTIX_DENOISER_AOV_TYPE_REFRACTION	
OPTIX_DENOISER_AOV_TYPE_DIFFUSE	

## 5.14.4.10 OptixDenoiserModelKind

enum [OptixDenoiserModelKind](#)

Model kind used by the denoiser.

See also [optixDenoiserCreate](#)

## Enumerator

OPTIX_DENOISER_MODEL_KIND_LDR	Use the built-in model appropriate for low dynamic range input.
OPTIX_DENOISER_MODEL_KIND_HDR	Use the built-in model appropriate for high dynamic range input.
OPTIX_DENOISER_MODEL_KIND_AOV	Use the built-in model appropriate for high dynamic range input and support for AOVs.
OPTIX_DENOISER_MODEL_KIND_TEMPORAL	Use the built-in model appropriate for high dynamic range input, temporally stable.
OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV	Use the built-in model appropriate for high dynamic range input and support for AOVs, temporally stable.
OPTIX_DENOISER_MODEL_KIND_UPSCALE2X	Use the built-in model appropriate for high dynamic range input and support for AOVs, upscaling 2x.
OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X	Use the built-in model appropriate for high dynamic range input and support for AOVs, upscaling 2x, temporally stable.

#### 5.14.4.11 OptixDeviceContextValidationMode

enum [OptixDeviceContextValidationMode](#)

Validation mode settings.

When enabled, certain device code utilities will be enabled to provide as good debug and error checking facilities as possible.

See also [optixDeviceContextCreate\(\)](#)

Enumerator

OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF
OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL

#### 5.14.4.12 OptixDeviceProperty

enum [OptixDeviceProperty](#)

Parameters used for [optixDeviceContextGetProperty\(\)](#)

See also [optixDeviceContextGetProperty\(\)](#)

Enumerator

OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH	Maximum value for <a href="#">OptixPipelineLinkOptions::maxTraceDepth</a> . sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH	Maximum value to pass into <a href="#">optixPipelineSetStackSize</a> for parameter <a href="#">maxTraversableGraphDepth</a> . sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS	The maximum number of primitives (over all build inputs) as input to a single Geometry Acceleration Structure (GAS). sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS	The maximum number of instances (over all build inputs) as input to a single Instance Acceleration Structure (IAS). sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_RTCORE_VERSION	The RT core version supported by the device (0 for no support, 10 for version 1.0). sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID	The maximum value for <a href="#">OptixInstance::instanceId</a> . sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK	The number of bits available for the <a href="#">OptixInstance::visibilityMask</a> . Higher bits must be set to zero. sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS	The maximum number of instances that can be added to a single Instance Acceleration Structure (IAS). sizeof(unsigned int)

## Enumerator

OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET	The maximum summed value of <code>OptixInstance::sbtOffset</code> . Also the maximum summed value of sbt offsets of all ancestor instances of a GAS in a traversable graph.
--	---

## 5.14.4.13 OptixDisplacementMicromapArrayIndexingMode

enum `OptixDisplacementMicromapArrayIndexingMode`

indexing mode of triangles to displacement micromaps in an array, used in `OptixBuildInputDisplacementMicromap`.

## Enumerator

OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_NONE	No displacement micromap is used.
OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_LINEAR	An implicit linear mapping of triangles to displacement micromaps in the displacement micromap array is used. <code>triangle[i]</code> will use <code>displacementMicromapArray[i]</code> .
OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_INDEXED	<code>OptixBuildInputDisplacementMicromap::displacementMicromapIndexBuffer</code> provides a per triangle array of indices into <code>OptixBuildInputDisplacementMicromap::displacementMicromapArray</code> . See <code>OptixBuildInputDisplacementMicromap::displacementMicromapIndexBuffer</code> for more details.

## 5.14.4.14 OptixDisplacementMicromapBiasAndScaleFormat

enum `OptixDisplacementMicromapBiasAndScaleFormat`

## Enumerator

OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_NONE
OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_FLOAT2
OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_HALF2

## 5.14.4.15 OptixDisplacementMicromapDirectionFormat

enum `OptixDisplacementMicromapDirectionFormat`

## Enumerator

OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_NONE
OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_FLOAT3
OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_HALF3



#### 5.14.4.16 OptixDisplacementMicromapFlags

enum `OptixDisplacementMicromapFlags`

Flags defining behavior of DMMs in a DMM array.

Enumerator

OPTIX_DISPLACEMENT_MICROMAP_FLAG_NONE	
OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_TRACE	This flag is mutually exclusive with OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_BUILD.
OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_BUILD	This flag is mutually exclusive with OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_TRACE.

#### 5.14.4.17 OptixDisplacementMicromapFormat

enum `OptixDisplacementMicromapFormat`

DMM input data format.

Enumerator

OPTIX_DISPLACEMENT_MICROMAP_FORMAT_NONE
OPTIX_DISPLACEMENT_MICROMAP_FORMAT_64_MICRO_TRIS_64_BYTES
OPTIX_DISPLACEMENT_MICROMAP_FORMAT_256_MICRO_TRIS_128_BYTES
OPTIX_DISPLACEMENT_MICROMAP_FORMAT_1024_MICRO_TRIS_128_BYTES

#### 5.14.4.18 OptixDisplacementMicromapTriangleFlags

enum `OptixDisplacementMicromapTriangleFlags`

Enumerator

OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_NONE	
OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_01	The triangle edge v0..v1 is decimated: after subdivision the number of micro triangles on that edge is halved such that a neighboring triangle can have a lower subdivision level without introducing cracks.
OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_12	The triangle edge v1..v2 is decimated.
OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_20	The triangle edge v2..v0 is decimated.

#### 5.14.4.19 OptixExceptionCodes

enum `OptixExceptionCodes`

The following values are used to indicate which exception was thrown.

#### Enumerator

OPTIX_EXCEPTION_CODE_STACK_OVERFLOW	Stack overflow of the continuation stack. no exception details.
OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED	The trace depth is exceeded. no exception details.
OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED	The traversal depth is exceeded. Exception details: <a href="#">optixGetTransformListSize()</a> <a href="#">optixGetTransformListHandle()</a>
OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE	Traversal encountered an invalid traversable type. Exception details: <a href="#">optixGetTransformListSize()</a> <a href="#">optixGetTransformListHandle()</a> <a href="#">optixGetExceptionInvalidTraversable()</a>
OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT	The miss SBT record index is out of bounds A miss SBT record index is valid within the range [0, <a href="#">OptixShaderBindingTable::missRecordCount</a> ) (See <a href="#">optixLaunch</a> ) Exception details: <a href="#">optixGetExceptionInvalidSbtOffset()</a>
OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT	The traversal hit SBT record index out of bounds. A traversal hit SBT record index is valid within the range [0, <a href="#">OptixShaderBindingTable::hitgroupRecordCount</a> ) (See <a href="#">optixLaunch</a> ) The following formula relates the sbt-geometry-acceleration-structure-index (See <a href="#">optixGetSbtGASIndex</a> ), sbt-stride-from-trace-call and sbt-offset-from-trace-call (See <a href="#">optixTrace</a> ) $\text{sbt-index} = \text{sbt-instance-offset} + (\text{sbt-geometry-acceleration-structure-index} * \text{sbt-stride-from-trace-call}) + \text{sbt-offset-from-trace-call}$ Exception details: <a href="#">optixGetTransformListSize()</a> <a href="#">optixGetTransformListHandle()</a> <a href="#">optixGetExceptionInvalidSbtOffset()</a> <a href="#">optixGetSbtGASIndex()</a>
OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE	The shader encountered an unsupported primitive type (See <a href="#">OptixPipelineCompileOptions::usesPrimitiveTypeFlags</a> ). no exception details.
OPTIX_EXCEPTION_CODE_INVALID_RAY	The shader encountered a call to <a href="#">optixTrace</a> with at least one of the float arguments being inf or nan, or the tmin argument is negative. Exception details: <a href="#">optixGetExceptionInvalidRay()</a>

## Enumerator

OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH	The shader encountered a call to either <code>optixDirectCall</code> or <code>optixCallableCall</code> where the argument count does not match the parameter count of the callable program which is called. Exception details: <code>optixGetExceptionParameterMismatch</code> .
OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH	The invoked builtin IS does not match the current GAS.
OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT	Tried to call a callable program using an SBT offset that is larger than the number of passed in callable SBT records. Exception details: <code>optixGetExceptionInvalidSbtOffset()</code>
OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD	Tried to call a direct callable using an SBT offset of a record that was built from a program group that did not include a direct callable.
OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD	Tried to call a continuation callable using an SBT offset of a record that was built from a program group that did not include a continuation callable.
OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS	Tried to directly traverse a single gas while single gas traversable graphs are not enabled (see <code>OptixTraversableGraphFlags::OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS</code> ). Exception details: <code>optixGetTransformListSize()</code> <code>optixGetTransformListHandle()</code> <code>optixGetExceptionInvalidTraversable()</code>
OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0	argument passed to an optix call is not within an acceptable range of values.
OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_1	
OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2	
OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS	Tried to access data on an AS without random data access support (See <code>OptixBuildFlags</code> ).
OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH	The program payload type doesn't match the trace payload type.

## 5.14.4.20 OptixExceptionFlags

enum `OptixExceptionFlags`

Exception flags.

See also `OptixPipelineCompileOptions::exceptionFlags`, `OptixExceptionCodes`

## Enumerator

OPTIX_EXCEPTION_FLAG_NONE	No exception are enabled.
---------------------------	---------------------------

## Enumerator

OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW	Enables exceptions check related to the continuation stack.
OPTIX_EXCEPTION_FLAG_TRACE_DEPTH	Enables exceptions check related to trace depth.
OPTIX_EXCEPTION_FLAG_USER	Enables user exceptions via <a href="#">optixThrowException()</a> . This flag must be specified for all modules in a pipeline if any module calls <a href="#">optixThrowException()</a> .
OPTIX_EXCEPTION_FLAG_DEBUG	Enables various exceptions check related to traversal.

## 5.14.4.21 OptixGeometryFlags

enum [OptixGeometryFlags](#)

Flags used by [OptixBuildInputTriangleArray::flags](#) and [OptixBuildInputCustomPrimitiveArray::flags](#).

## Enumerator

OPTIX_GEOMETRY_FLAG_NONE	No flags set.
OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT	Disables the invocation of the anyhit program. Can be overridden by OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT and OPTIX_RAY_FLAG_ENFORCE_ANYHIT.
OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL	If set, an intersection with the primitive will trigger one and only one invocation of the anyhit program. Otherwise, the anyhit program may be invoked more than once.
OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING	Prevent triangles from getting culled due to their orientation. Effectively ignores ray flags OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES and OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES.

## 5.14.4.22 OptixHitKind

enum [OptixHitKind](#)

Legacy type: A subset of the hit kinds for built-in primitive intersections. It is preferred to use [optixGetPrimitiveType\(\)](#), together with [optixIsFrontFaceHit\(\)](#) or [optixIsBackFaceHit\(\)](#).

See also [optixGetHitKind\(\)](#)

## Enumerator

OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE	Ray hit the triangle on the front face.
OPTIX_HIT_KIND_TRIANGLE_BACK_FACE	Ray hit the triangle on the back face.

#### 5.14.4.23 OptixIndicesFormat

enum `OptixIndicesFormat`

Format of indices used in `OptixBuildInputTriangleArray::indexFormat`.

Enumerator

<code>OPTIX_INDICES_FORMAT_NONE</code>	No indices, this format must only be used in combination with triangle soups, i.e., <code>numIndexTriplets</code> must be zero.
<code>OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3</code>	Three shorts.
<code>OPTIX_INDICES_FORMAT_UNSIGNED_INT3</code>	Three ints.

#### 5.14.4.24 OptixInstanceFlags

enum `OptixInstanceFlags`

Flags set on the `OptixInstance::flags`.

These can be or'ed together to combine multiple flags.

Enumerator

<code>OPTIX_INSTANCE_FLAG_NONE</code>	No special flag set.
<code>OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING</code>	Prevent triangles from getting culled due to their orientation. Effectively ignores ray flags <code>OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES</code> and <code>OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES</code> .
<code>OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING</code>	Flip triangle orientation. This affects front/backface culling as well as the reported face in case of a hit.
<code>OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT</code>	Disable anyhit programs for all geometries of the instance. Can be overridden by <code>OPTIX_RAY_FLAG_ENFORCE_ANYHIT</code> . This flag is mutually exclusive with <code>OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT</code> .
<code>OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT</code>	Enables anyhit programs for all geometries of the instance. Overrides <code>OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT</code> . Can be overridden by <code>OPTIX_RAY_FLAG_DISABLE_ANYHIT</code> . This flag is mutually exclusive with <code>OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT</code> .
<code>OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE</code>	Force 4-state opacity micromaps to behave as 2-state opacity micromaps during traversal.
<code>OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS</code>	Don't perform opacity micromap query for this instance. GAS must be built with <code>ALLOW_DISABLE_OPACITY_MICROMAPS</code> for this to be valid. This flag overrides <code>FORCE_OPACITY_MICROMAP_2_STATE</code> instance and ray flags.

#### 5.14.4.25 OptixModuleCompileState

enum [OptixModuleCompileState](#)

Module compilation state.

See also [optixModuleGetCompilationState\(\)](#), [optixModuleCreateWithTasks\(\)](#)

Enumerator

OPTIX_MODULE_COMPILE_STATE_NOT_STARTED	No OptixTask objects have started.
OPTIX_MODULE_COMPILE_STATE_STARTED	Started, but not all OptixTask objects have completed. No detected failures.
OPTIX_MODULE_COMPILE_STATE_PENDING_FAILURE	Not all OptixTask objects have completed, but at least one has failed.
OPTIX_MODULE_COMPILE_STATE_FAILED	All OptixTask objects have completed, and at least one has failed.
OPTIX_MODULE_COMPILE_STATE_COMPLETED	All OptixTask objects have completed. The OptixModule is ready to be used.

#### 5.14.4.26 OptixMotionFlags

enum [OptixMotionFlags](#)

Enum to specify motion flags.

See also [OptixMotionOptions::flags](#).

Enumerator

OPTIX_MOTION_FLAG_NONE
OPTIX_MOTION_FLAG_START_VANISH
OPTIX_MOTION_FLAG_END_VANISH

#### 5.14.4.27 OptixOpacityMicromapArrayIndexingMode

enum [OptixOpacityMicromapArrayIndexingMode](#)

indexing mode of triangles to opacity micromaps in an array, used in [OptixBuildInputOpacityMicromap](#).

Enumerator

OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE	No opacity micromap is used.
OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR	An implicit linear mapping of triangles to opacity micromaps in the opacity micromap array is used. triangle[i] will use opacityMicromapArray[i].

## Enumerator

OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED	<a href="#">OptixBuildInputOpacityMicromap::indexBuffer</a> provides a per triangle array of predefined indices and/or indices into <a href="#">OptixBuildInputOpacityMicromap::opacityMicromapArray</a> . See <a href="#">OptixBuildInputOpacityMicromap::indexBuffer</a> for more details.
--	--

## 5.14.4.28 OptixOpacityMicromapFlags

enum [OptixOpacityMicromapFlags](#)

Flags defining behavior of opacity micromaps in a opacity micromap array.

## Enumerator

OPTIX_OPACITY_MICROMAP_FLAG_NONE	
OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE	This flag is mutually exclusive with OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD.
OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD	This flag is mutually exclusive with OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE.

## 5.14.4.29 OptixOpacityMicromapFormat

enum [OptixOpacityMicromapFormat](#)

Specifies whether to use a 2- or 4-state opacity micromap format.

## Enumerator

OPTIX_OPACITY_MICROMAP_FORMAT_NONE	invalid format
OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE	0: Transparent, 1: Opaque
OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE	0: Transparent, 1: Opaque, 2: Unknown-Transparent, 3: Unknown-Opaque

## 5.14.4.30 OptixPayloadSemantics

enum [OptixPayloadSemantics](#)

Semantic flags for a single payload word.

Used to specify the semantics of a payload word per shader type. "read": Shader of this type may read the payload word. "write": Shader of this type may write the payload word.

"trace\_caller\_write": Shaders may consume the value of the payload word passed to [optixTrace](#) by the caller. "trace\_caller\_read": The caller to [optixTrace](#) may read the payload word after the call to [optixTrace](#).

Semantics can be bitwise combined. Combining "read" and "write" is equivalent to specifying "read\_

write". A payload needs to be writable by the caller or at least one shader type. A payload needs to be readable by the caller or at least one shader type after a being writable.

Enumerator

OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE
OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ
OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE
OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE
OPTIX_PAYLOAD_SEMANTICS_CH_NONE
OPTIX_PAYLOAD_SEMANTICS_CH_READ
OPTIX_PAYLOAD_SEMANTICS_CH_WRITE
OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE
OPTIX_PAYLOAD_SEMANTICS_MS_NONE
OPTIX_PAYLOAD_SEMANTICS_MS_READ
OPTIX_PAYLOAD_SEMANTICS_MS_WRITE
OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE
OPTIX_PAYLOAD_SEMANTICS_AH_NONE
OPTIX_PAYLOAD_SEMANTICS_AH_READ
OPTIX_PAYLOAD_SEMANTICS_AH_WRITE
OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE
OPTIX_PAYLOAD_SEMANTICS_IS_NONE
OPTIX_PAYLOAD_SEMANTICS_IS_READ
OPTIX_PAYLOAD_SEMANTICS_IS_WRITE
OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE

#### 5.14.4.31 OptixPayloadTypeID

enum `OptixPayloadTypeID`

Payload type identifiers.

Enumerator

OPTIX_PAYLOAD_TYPE_DEFAULT	
OPTIX_PAYLOAD_TYPE_ID_0	
OPTIX_PAYLOAD_TYPE_ID_1	
OPTIX_PAYLOAD_TYPE_ID_2	
OPTIX_PAYLOAD_TYPE_ID_3	
OPTIX_PAYLOAD_TYPE_ID_4	
OPTIX_PAYLOAD_TYPE_ID_5	
OPTIX_PAYLOAD_TYPE_ID_6	
OPTIX_PAYLOAD_TYPE_ID_7	



### 5.14.4.32 OptixPixelFormat

enum [OptixPixelFormat](#)

Pixel formats used by the denoiser.

See also [OptixImage2D::format](#)

Enumerator

OPTIX_PIXEL_FORMAT_HALF1	one half
OPTIX_PIXEL_FORMAT_HALF2	two halves, XY
OPTIX_PIXEL_FORMAT_HALF3	three halves, RGB
OPTIX_PIXEL_FORMAT_HALF4	four halves, RGBA
OPTIX_PIXEL_FORMAT_FLOAT1	one float
OPTIX_PIXEL_FORMAT_FLOAT2	two floats, XY
OPTIX_PIXEL_FORMAT_FLOAT3	three floats, RGB
OPTIX_PIXEL_FORMAT_FLOAT4	four floats, RGBA
OPTIX_PIXEL_FORMAT_UCHAR3	three unsigned chars, RGB
OPTIX_PIXEL_FORMAT_UCHAR4	four unsigned chars, RGBA
OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER	internal format

### 5.14.4.33 OptixPrimitiveType

enum [OptixPrimitiveType](#)

Builtin primitive types.

Enumerator

OPTIX_PRIMITIVE_TYPE_CUSTOM	Custom primitive.
OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE	B-spline curve of degree 2 with circular cross-section.
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE	B-spline curve of degree 3 with circular cross-section.
OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR	Piecewise linear curve with circular cross-section.
OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM	CatmullRom curve with circular cross-section.
OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE	B-spline curve of degree 2 with oriented, flat cross-section.
OPTIX_PRIMITIVE_TYPE_SPHERE	Sphere.
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER	Bezier curve of degree 3 with circular cross-section.
OPTIX_PRIMITIVE_TYPE_TRIANGLE	Triangle.
OPTIX_PRIMITIVE_TYPE_DISPLACED_MICROMESH_TRIANGLE	Triangle with an applied displacement micromap.

#### 5.14.4.34 OptixPrimitiveTypeFlags

enum [OptixPrimitiveTypeFlags](#)

Builtin flags may be bitwise combined.

See also [OptixPipelineCompileOptions::usesPrimitiveTypeFlags](#)

Enumerator

OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM	Custom primitive.
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE	B-spline curve of degree 2 with circular cross-section.
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE	B-spline curve of degree 3 with circular cross-section.
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR	Piecewise linear curve with circular cross-section.
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM	CatmullRom curve with circular cross-section.
OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE	B-spline curve of degree 2 with oriented, flat cross-section.
OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE	Sphere.
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER	Bezier curve of degree 3 with circular cross-section.
OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE	Triangle.
OPTIX_PRIMITIVE_TYPE_FLAGS_DISPLACED_MICROMESH_TRIANGLE	Triangle with an applied displacement micromap.

#### 5.14.4.35 OptixProgramGroupFlags

enum [OptixProgramGroupFlags](#)

Flags for program groups.

Enumerator

OPTIX_PROGRAM_GROUP_FLAGS_NONE	Currently there are no flags.
--------------------------------	-------------------------------

#### 5.14.4.36 OptixProgramGroupKind

enum [OptixProgramGroupKind](#)

Distinguishes different kinds of program groups.

Enumerator

OPTIX_PROGRAM_GROUP_KIND_RAYGEN	Program group containing a raygen (RG) program. See also <a href="#">OptixProgramGroupSingleModule</a> , <a href="#">OptixProgramGroupDesc::raygen</a>
---------------------------------	--

## Enumerator

OPTIX_PROGRAM_GROUP_KIND_MISS	Program group containing a miss (MS) program. See also <a href="#">OptixProgramGroupSingleModule</a> , <a href="#">OptixProgramGroupDesc::miss</a>
OPTIX_PROGRAM_GROUP_KIND_EXCEPTION	Program group containing an exception (EX) program. See also <a href="#">OptixProgramGroupHitgroup</a> , <a href="#">OptixProgramGroupDesc::exception</a>
OPTIX_PROGRAM_GROUP_KIND_HITGROUP	Program group containing an intersection (IS), any hit (AH), and/or closest hit (CH) program. See also <a href="#">OptixProgramGroupSingleModule</a> , <a href="#">OptixProgramGroupDesc::hitgroup</a>
OPTIX_PROGRAM_GROUP_KIND_CALLABLES	Program group containing a direct (DC) or continuation (CC) callable program. See also <a href="#">OptixProgramGroupCallables</a> , <a href="#">OptixProgramGroupDesc::callables</a>

## 5.14.4.37 OptixQueryFunctionTableOptions

enum [OptixQueryFunctionTableOptions](#)

Options that can be passed to [optixQueryFunctionTable\(\)](#)

## Enumerator

OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY	Placeholder (there are no options yet)
---	--

## 5.14.4.38 OptixRayFlags

enum [OptixRayFlags](#)

Ray flags passed to the device function [optixTrace\(\)](#). These affect the behavior of traversal per invocation.

See also [optixTrace\(\)](#)

## Enumerator

OPTIX_RAY_FLAG_NONE	No change from the behavior configured for the individual AS.
OPTIX_RAY_FLAG_DISABLE_ANYHIT	Disables anyhit programs for the ray. Overrides OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT. This flag is mutually exclusive with OPTIX_RAY_FLAG_ENFORCE_ANYHIT, OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT, OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT.

## Enumerator

OPTIX_RAY_FLAG_ENFORCE_ANYHIT	Forces anyhit program execution for the ray. Overrides OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT as well as OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT. This flag is mutually exclusive with OPTIX_RAY_FLAG_DISABLE_ANYHIT, OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT, OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT.
OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT	Terminates the ray after the first hit and executes the closesthit program of that hit.
OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT	Disables closesthit programs for the ray, but still executes miss program in case of a miss.
OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES	Do not intersect triangle back faces (respects a possible face change due to instance flag OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING). This flag is mutually exclusive with OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES.
OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES	Do not intersect triangle front faces (respects a possible face change due to instance flag OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING). This flag is mutually exclusive with OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES.
OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT	Do not intersect geometry which disables anyhit programs (due to setting geometry flag OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT or instance flag OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT). This flag is mutually exclusive with OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT, OPTIX_RAY_FLAG_ENFORCE_ANYHIT, OPTIX_RAY_FLAG_DISABLE_ANYHIT.
OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT	Do not intersect geometry which have an enabled anyhit program (due to not setting geometry flag OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT or setting instance flag OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT). This flag is mutually exclusive with OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT, OPTIX_RAY_FLAG_ENFORCE_ANYHIT, OPTIX_RAY_FLAG_DISABLE_ANYHIT.
OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE	Force 4-state opacity micromaps to behave as 2-state opacity micromaps during traversal.

## 5.14.4.39 OptixResult

enum `OptixResult`

Result codes returned from API functions.

All host side API functions return `OptixResult` with the exception of `optixGetErrorName` and `optixGetErrorString`. When successful `OPTIX_SUCCESS` is returned. All return codes except for `OPTIX_SUCCESS` should be assumed to be errors as opposed to a warning.

See also `optixGetErrorName()`, `optixGetErrorString()`

Enumerator

OPTIX_SUCCESS
OPTIX_ERROR_INVALID_VALUE
OPTIX_ERROR_HOST_OUT_OF_MEMORY
OPTIX_ERROR_INVALID_OPERATION
OPTIX_ERROR_FILE_IO_ERROR
OPTIX_ERROR_INVALID_FILE_FORMAT
OPTIX_ERROR_DISK_CACHE_INVALID_PATH
OPTIX_ERROR_DISK_CACHE_PERMISSION_ERROR
OPTIX_ERROR_DISK_CACHE_DATABASE_ERROR
OPTIX_ERROR_DISK_CACHE_INVALID_DATA
OPTIX_ERROR_LAUNCH_FAILURE
OPTIX_ERROR_INVALID_DEVICE_CONTEXT
OPTIX_ERROR_CUDA_NOT_INITIALIZED
OPTIX_ERROR_VALIDATION_FAILURE
OPTIX_ERROR_INVALID_INPUT
OPTIX_ERROR_INVALID_LAUNCH_PARAMETER
OPTIX_ERROR_INVALID_PAYLOAD_ACCESS
OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS
OPTIX_ERROR_INVALID_FUNCTION_USE
OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS
OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY
OPTIX_ERROR_PIPELINE_LINK_ERROR
OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE
OPTIX_ERROR_INTERNAL_COMPILER_ERROR
OPTIX_ERROR_DENOISER_MODEL_NOT_SET
OPTIX_ERROR_DENOISER_NOT_INITIALIZED
OPTIX_ERROR_NOT_COMPATIBLE
OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH
OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED
OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID
OPTIX_ERROR_NOT_SUPPORTED
OPTIX_ERROR_UNSUPPORTED_ABI_VERSION
OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH

## Enumerator

OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS
OPTIX_ERROR_LIBRARY_NOT_FOUND
OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND
OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE
OPTIX_ERROR_DEVICE_OUT_OF_MEMORY
OPTIX_ERROR_CUDA_ERROR
OPTIX_ERROR_INTERNAL_ERROR
OPTIX_ERROR_UNKNOWN

## 5.14.4.40 OptixTransformFormat

enum [OptixTransformFormat](#)

Format of transform used in [OptixBuildInputTriangleArray::transformFormat](#).

## Enumerator

OPTIX_TRANSFORM_FORMAT_NONE	no transform, default for zero initialization
OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12	3x4 row major affine matrix

## 5.14.4.41 OptixTransformType

enum [OptixTransformType](#)

Transform.

[OptixTransformType](#) is used by the device function [optixGetTransformTypeFromHandle\(\)](#) to determine the type of the [OptixTraversableHandle](#) returned from [optixGetTransformListHandle\(\)](#).

## Enumerator

OPTIX_TRANSFORM_TYPE_NONE	Not a transformation.
OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM	See also <a href="#">OptixStaticTransform</a>
OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM	See also <a href="#">OptixMatrixMotionTransform</a>
OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM	See also <a href="#">OptixSRTMotionTransform</a>
OPTIX_TRANSFORM_TYPE_INSTANCE	See also <a href="#">OptixInstance</a>

## 5.14.4.42 OptixTraversableGraphFlags

enum [OptixTraversableGraphFlags](#)

Specifies the set of valid traversable graphs that may be passed to invocation of [optixTrace\(\)](#). Flags may be bitwise combined.

## Enumerator

OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY	Used to signal that any traversable graphs is valid. This flag is mutually exclusive with all other flags.
OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS	Used to signal that a traversable graph of a single Geometry Acceleration Structure (GAS) without any transforms is valid. This flag may be combined with other flags except for OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY.
OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING	Used to signal that a traversable graph of a single Instance Acceleration Structure (IAS) directly connected to Geometry Acceleration Structure (GAS) traversables without transform traversables in between is valid. This flag may be combined with other flags except for OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY.

## 5.14.4.43 OptixTraversableType

enum `OptixTraversableType`

Traversable Handles.

See also `optixConvertPointerToTraversableHandle()`

## Enumerator

OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM	Static transforms. See also <a href="#">OptixStaticTransform</a>
OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM	Matrix motion transform. See also <a href="#">OptixMatrixMotionTransform</a>
OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM	SRT motion transform. See also <a href="#">OptixSRTMotionTransform</a>

## 5.14.4.44 OptixVertexFormat

enum `OptixVertexFormat`Format of vertices used in `OptixBuildInputTriangleArray::vertexFormat`.

## Enumerator

OPTIX_VERTEX_FORMAT_NONE	No vertices.
OPTIX_VERTEX_FORMAT_FLOAT3	Vertices are represented by three floats.
OPTIX_VERTEX_FORMAT_FLOAT2	Vertices are represented by two floats.
OPTIX_VERTEX_FORMAT_HALF3	Vertices are represented by three halves.
OPTIX_VERTEX_FORMAT_HALF2	Vertices are represented by two halves.
OPTIX_VERTEX_FORMAT_SNORM16_3	

## Enumerator

OPTIX_VERTEX_FORMAT_SNORM16_2	
-------------------------------	--

## 6 Namespace Documentation

### 6.1 optix\_impl Namespace Reference

#### Functions

- static \_\_forceinline\_\_ \_\_device\_\_ void [optixDumpStaticTransformFromHandle](#) ([OptixTraversableHandle](#) handle)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixDumpMotionMatrixTransformFromHandle](#) ([OptixTraversableHandle](#) handle)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixDumpSrtMatrixTransformFromHandle](#) ([OptixTraversableHandle](#) handle)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixDumpInstanceFromHandle](#) ([OptixTraversableHandle](#) handle)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixDumpTransform](#) ([OptixTraversableHandle](#) handle)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixDumpTransformList](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixDumpExceptionDetails](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ float4 [optixAddFloat4](#) (const float4 &a, const float4 &b)
- static \_\_forceinline\_\_ \_\_device\_\_ float4 [optixMulFloat4](#) (const float4 &a, float b)
- static \_\_forceinline\_\_ \_\_device\_\_ uint4 [optixLdg](#) (unsigned long long addr)
- template<class T >  
static \_\_forceinline\_\_ \_\_device\_\_ T [optixLoadReadOnlyAlign16](#) (const T \*ptr)
- static \_\_forceinline\_\_ \_\_device\_\_ float4 [optixMultiplyRowMatrix](#) (const float4 vec, const float4 m0, const float4 m1, const float4 m2)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixGetMatrixFromSrt](#) (float4 &m0, float4 &m1, float4 &m2, const [OptixSRTData](#) &srt)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixInvertMatrix](#) (float4 &m0, float4 &m1, float4 &m2)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixLoadInterpolatedMatrixKey](#) (float4 &m0, float4 &m1, float4 &m2, const float4 \*matrix, const float t1)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixLoadInterpolatedSrtKey](#) (float4 &srt0, float4 &srt1, float4 &srt2, float4 &srt3, const float4 \*srt, const float t1)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixResolveMotionKey](#) (float &localt, int &key, const [OptixMotionOptions](#) &options, const float globalt)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixGetInterpolatedTransformation](#) (float4 &trf0, float4 &trf1, float4 &trf2, const [OptixMatrixMotionTransform](#) \*transformData, const float time)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixGetInterpolatedTransformation](#) (float4 &trf0, float4 &trf1, float4 &trf2, const [OptixSRTMotionTransform](#) \*transformData, const float time)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixGetInterpolatedTransformationFromHandle](#) (float4 &trf0, float4 &trf1, float4 &trf2, const [OptixTraversableHandle](#) handle, const float time, const bool objectToWorld)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixGetWorldToObjectTransformMatrix](#) (float4 &m0, float4 &m1, float4 &m2)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixGetObjectToWorldTransformMatrix](#) (float4 &m0, float4 &m1, float4 &m2)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 [optixTransformPoint](#) (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &p)



- static `__forceinline__ __device__ float3` [optixTransformVector](#) (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &v)
- static `__forceinline__ __device__ float3` [optixTransformNormal](#) (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &n)

## 6.1.1 Function Documentation

### 6.1.1.1 optixAddFloat4()

```
static __forceinline__ __device__ float4 optix_impl::optixAddFloat4 (
    const float4 & a,
    const float4 & b ) [static]
```

### 6.1.1.2 optixDumpExceptionDetails()

```
static __forceinline__ __device__ void optix_impl::optixDumpExceptionDetails
( ) [static]
```

### 6.1.1.3 optixDumpInstanceFromHandle()

```
static __forceinline__ __device__ void optix_impl
::optixDumpInstanceFromHandle (
    OptixTraversableHandle handle ) [static]
```

### 6.1.1.4 optixDumpMotionMatrixTransformFromHandle()

```
static __forceinline__ __device__ void optix_impl
::optixDumpMotionMatrixTransformFromHandle (
    OptixTraversableHandle handle ) [static]
```

### 6.1.1.5 optixDumpSrtMatrixTransformFromHandle()

```
static __forceinline__ __device__ void optix_impl
::optixDumpSrtMatrixTransformFromHandle (
    OptixTraversableHandle handle ) [static]
```

### 6.1.1.6 optixDumpStaticTransformFromHandle()

```
static __forceinline__ __device__ void optix_impl
::optixDumpStaticTransformFromHandle (
    OptixTraversableHandle handle ) [static]
```

### 6.1.1.7 optixDumpTransform()

```
static __forceinline__ __device__ void optix_impl::optixDumpTransform (
    OptixTraversableHandle handle ) [static]
```

### 6.1.1.8 optixDumpTransformList()

```
static __forceinline__ __device__ void optix_impl::optixDumpTransformList (
) [static]
```

## 6.1.1.9 optixGetInterpolatedTransformation() [1/2]

```
static __forceinline__ __device__ void optix_impl
::optixGetInterpolatedTransformation (
    float4 & trf0,
    float4 & trf1,
    float4 & trf2,
    const OptixMatrixMotionTransform * transformData,
    const float time ) [static]
```

## 6.1.1.10 optixGetInterpolatedTransformation() [2/2]

```
static __forceinline__ __device__ void optix_impl
::optixGetInterpolatedTransformation (
    float4 & trf0,
    float4 & trf1,
    float4 & trf2,
    const OptixSRTMotionTransform * transformData,
    const float time ) [static]
```

## 6.1.1.11 optixGetInterpolatedTransformationFromHandle()

```
static __forceinline__ __device__ void optix_impl
::optixGetInterpolatedTransformationFromHandle (
    float4 & trf0,
    float4 & trf1,
    float4 & trf2,
    const OptixTraversableHandle handle,
    const float time,
    const bool objectToWorld ) [static]
```

## 6.1.1.12 optixGetMatrixFromSrt()

```
static __forceinline__ __device__ void optix_impl::optixGetMatrixFromSrt (
    float4 & m0,
    float4 & m1,
    float4 & m2,
    const OptixSRTData & srt ) [static]
```

## 6.1.1.13 optixGetObjectToWorldTransformMatrix()

```
static __forceinline__ __device__ void optix_impl
::optixGetObjectToWorldTransformMatrix (
    float4 & m0,
    float4 & m1,
    float4 & m2 ) [static]
```

## 6.1.1.14 optixGetWorldToObjectTransformMatrix()

```
static __forceinline__ __device__ void optix_impl::optixGetWorldToObjectTransformMatrix (
    float4 & m0,
    float4 & m1,
    float4 & m2 ) [static]
```

## 6.1.1.15 optixInvertMatrix()

```
static __forceinline__ __device__ void optix_impl::optixInvertMatrix (
    float4 & m0,
    float4 & m1,
    float4 & m2 ) [static]
```

## 6.1.1.16 optixLdg()

```
static __forceinline__ __device__ uint4 optix_impl::optixLdg (
    unsigned long long addr ) [static]
```

## 6.1.1.17 optixLoadInterpolatedMatrixKey()

```
static __forceinline__ __device__ void optix_impl::optixLoadInterpolatedMatrixKey (
    float4 & m0,
    float4 & m1,
    float4 & m2,
    const float4 * matrix,
    const float t1 ) [static]
```

## 6.1.1.18 optixLoadInterpolatedSrtKey()

```
static __forceinline__ __device__ void optix_impl::optixLoadInterpolatedSrtKey (
    float4 & srt0,
    float4 & srt1,
    float4 & srt2,
    float4 & srt3,
    const float4 * srt,
    const float t1 ) [static]
```

## 6.1.1.19 optixLoadReadOnlyAlign16()

```
template<class T >
static __forceinline__ __device__ T optix_impl::optixLoadReadOnlyAlign16 (
    const T * ptr ) [static]
```

## 6.1.1.20 optixMulFloat4()

```
static __forceinline__ __device__ float4 optix_impl::optixMulFloat4 (
    const float4 & a,
    float b ) [static]
```

## 6.1.1.21 optixMultiplyRowMatrix()

```
static __forceinline__ __device__ float4 optix_impl::optixMultiplyRowMatrix
(
    const float4 vec,
    const float4 m0,
    const float4 m1,
    const float4 m2 ) [static]
```

## 6.1.1.22 optixResolveMotionKey()

```
static __forceinline__ __device__ void optix_impl::optixResolveMotionKey (
    float & localt,
    int & key,
    const OptixMotionOptions & options,
    const float globalt ) [static]
```

## 6.1.1.23 optixTransformNormal()

```
static __forceinline__ __device__ float3 optix_impl::optixTransformNormal (
    const float4 & m0,
    const float4 & m1,
    const float4 & m2,
    const float3 & n ) [static]
```

## 6.1.1.24 optixTransformPoint()

```
static __forceinline__ __device__ float3 optix_impl::optixTransformPoint (
    const float4 & m0,
    const float4 & m1,
    const float4 & m2,
    const float3 & p ) [static]
```

## 6.1.1.25 optixTransformVector()

```
static __forceinline__ __device__ float3 optix_impl::optixTransformVector (
    const float4 & m0,
    const float4 & m1,
    const float4 & m2,
    const float3 & v ) [static]
```

## 6.2 optix\_internal Namespace Reference

### Classes

- struct [TypePack](#)

## 7 Class Documentation

### 7.1 OptixAabb Struct Reference

```
#include <optix_types.h>
```

#### Public Attributes

- float [minX](#)
- float [minY](#)
- float [minZ](#)
- float [maxX](#)
- float [maxY](#)
- float [maxZ](#)

#### 7.1.1 Detailed Description

AABB inputs.

#### 7.1.2 Member Data Documentation

##### 7.1.2.1 [maxX](#)

float [OptixAabb::maxX](#)

Upper extent in X direction.

##### 7.1.2.2 [maxY](#)

float [OptixAabb::maxY](#)

Upper extent in Y direction.

##### 7.1.2.3 [maxZ](#)

float [OptixAabb::maxZ](#)

Upper extent in Z direction.

##### 7.1.2.4 [minX](#)

float [OptixAabb::minX](#)

Lower extent in X direction.

##### 7.1.2.5 [minY](#)

float [OptixAabb::minY](#)

Lower extent in Y direction.

### 7.1.2.6 minZ

float OptixAabb::minZ

Lower extent in Z direction.

## 7.2 OptixAccelBufferSizes Struct Reference

#include <optix\_types.h>

### Public Attributes

- size\_t [outputSizeInBytes](#)
- size\_t [tempSizeInBytes](#)
- size\_t [tempUpdateSizeInBytes](#)

### 7.2.1 Detailed Description

Struct for querying builder allocation requirements.

Once queried the sizes should be used to allocate device memory of at least these sizes.

See also [optixAccelComputeMemoryUsage\(\)](#)

### 7.2.2 Member Data Documentation

#### 7.2.2.1 outputSizeInBytes

size\_t OptixAccelBufferSizes::outputSizeInBytes

The size in bytes required for the outputBuffer parameter to optixAccelBuild when doing a build (OPTIX\_BUILD\_OPERATION\_BUILD).

#### 7.2.2.2 tempSizeInBytes

size\_t OptixAccelBufferSizes::tempSizeInBytes

The size in bytes required for the tempBuffer paramter to optixAccelBuild when doing a build (OPTIX\_BUILD\_OPERATION\_BUILD).

#### 7.2.2.3 tempUpdateSizeInBytes

size\_t OptixAccelBufferSizes::tempUpdateSizeInBytes

The size in bytes required for the tempBuffer parameter to optixAccelBuild when doing an update (OPTIX\_BUILD\_OPERATION\_UPDATE). This value can be different than tempSizeInBytes used for a full build. Only non-zero if OPTIX\_BUILD\_FLAG\_ALLOW\_UPDATE flag is set in [OptixAccelBuildOptions](#).

## 7.3 OptixAccelBuildOptions Struct Reference

#include <optix\_types.h>

### Public Attributes

- unsigned int [buildFlags](#)
- [OptixBuildOperation](#) operation
- [OptixMotionOptions](#) motionOptions

### 7.3.1 Detailed Description

Build options for acceleration structures.

See also [optixAccelComputeMemoryUsage\(\)](#), [optixAccelBuild\(\)](#)

### 7.3.2 Member Data Documentation

#### 7.3.2.1 buildFlags

`unsigned int OptixAccelBuildOptions::buildFlags`

Combinations of OptixBuildFlags.

#### 7.3.2.2 motionOptions

`OptixMotionOptions OptixAccelBuildOptions::motionOptions`

Options for motion.

#### 7.3.2.3 operation

`OptixBuildOperation OptixAccelBuildOptions::operation`

If `OPTIX_BUILD_OPERATION_UPDATE` the output buffer is assumed to contain the result of a full build with `OPTIX_BUILD_FLAG_ALLOW_UPDATE` set and using the same number of primitives. It is updated incrementally to reflect the current position of the primitives. If a BLAS has been built with `OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE`, new opacity micromap arrays and opacity micromap indices may be provided to the refit.

## 7.4 OptixAccelEmitDesc Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- `CUdeviceptr result`
- `OptixAccelPropertyType type`

### 7.4.1 Detailed Description

Specifies a type and output destination for emitted post-build properties.

See also [optixAccelBuild\(\)](#)

### 7.4.2 Member Data Documentation

#### 7.4.2.1 result

`CUdeviceptr OptixAccelEmitDesc::result`

Output buffer for the properties.

#### 7.4.2.2 type

`OptixAccelPropertyType OptixAccelEmitDesc::type`

Requested property.

## 7.5 OptixBuildInput Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- [OptixBuildInputType](#) type
- union {
  - [OptixBuildInputTriangleArray](#) triangleArray
  - [OptixBuildInputCurveArray](#) curveArray
  - [OptixBuildInputSphereArray](#) sphereArray
  - [OptixBuildInputCustomPrimitiveArray](#) customPrimitiveArray
  - [OptixBuildInputInstanceArray](#) instanceArray
  - char pad [1024]
- };

### 7.5.1 Detailed Description

Build inputs.

All of them support motion and the size of the data arrays needs to match the number of motion steps

See also [optixAccelComputeMemoryUsage\(\)](#), [optixAccelBuild\(\)](#)

### 7.5.2 Member Data Documentation

#### 7.5.2.1

```
union { ... } OptixBuildInput::@1
```

#### 7.5.2.2 [curveArray](#)

```
OptixBuildInputCurveArray OptixBuildInput::curveArray
```

Curve inputs.

#### 7.5.2.3 [customPrimitiveArray](#)

```
OptixBuildInputCustomPrimitiveArray OptixBuildInput::customPrimitiveArray
```

Custom primitive inputs.

#### 7.5.2.4 [instanceArray](#)

```
OptixBuildInputInstanceArray OptixBuildInput::instanceArray
```

Instance and instance pointer inputs.

#### 7.5.2.5 [pad](#)

```
char OptixBuildInput::pad[1024]
```

#### 7.5.2.6 [sphereArray](#)

```
OptixBuildInputSphereArray OptixBuildInput::sphereArray
```

Sphere inputs.



### 7.5.2.7 triangleArray

[OptixBuildInputTriangleArray](#) [OptixBuildInput::triangleArray](#)

Triangle inputs.

### 7.5.2.8 type

[OptixBuildInputType](#) [OptixBuildInput::type](#)

The type of the build input.

## 7.6 OptixBuildInputCurveArray Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- [OptixPrimitiveType](#) [curveType](#)
- unsigned int [numPrimitives](#)
- const [CUdeviceptr](#) \* [vertexBuffers](#)
- unsigned int [numVertices](#)
- unsigned int [vertexStrideInBytes](#)
- const [CUdeviceptr](#) \* [widthBuffers](#)
- unsigned int [widthStrideInBytes](#)
- const [CUdeviceptr](#) \* [normalBuffers](#)
- unsigned int [normalStrideInBytes](#)
- [CUdeviceptr](#) [indexBuffer](#)
- unsigned int [indexStrideInBytes](#)
- unsigned int [flag](#)
- unsigned int [primitiveIndexOffset](#)
- unsigned int [endcapFlags](#)

### 7.6.1 Detailed Description

Curve inputs.

A curve is a swept surface defined by a 3D spline curve and a varying width (radius). A curve (or "strand") of degree  $d$  ( $3=\text{cubic}$ ,  $2=\text{quadratic}$ ,  $1=\text{linear}$ ) is represented by  $N > d$  vertices and  $N$  width values, and comprises  $N - d$  segments. Each segment is defined by  $d+1$  consecutive vertices. Each curve may have a different number of vertices.

OptiX describes the curve array as a list of curve segments. The primitive id is the segment number. It is the user's responsibility to maintain a mapping between curves and curve segments. Each index buffer entry  $i = \text{indexBuffer}[\text{primid}]$  specifies the start of a curve segment, represented by  $d+1$  consecutive vertices in the vertex buffer, and  $d+1$  consecutive widths in the width buffer. Width is interpolated the same way vertices are interpolated, that is, using the curve basis.

Each curves build input has only one SBT record. To create curves with different materials in the same BVH, use multiple build inputs.

See also [OptixBuildInput::curveArray](#)

### 7.6.2 Member Data Documentation

#### 7.6.2.1 curveType

[OptixPrimitiveType](#) [OptixBuildInputCurveArray::curveType](#)

Curve degree and basis.

See also [OptixPrimitiveType](#)

### 7.6.2.2 endcapFlags

`unsigned int OptixBuildInputCurveArray::endcapFlags`

End cap flags, see [OptixCurveEndcapFlags](#).

### 7.6.2.3 flag

`unsigned int OptixBuildInputCurveArray::flag`

Combination of [OptixGeometryFlags](#) describing the primitive behavior.

### 7.6.2.4 indexBuffer

`CUdeviceptr OptixBuildInputCurveArray::indexBuffer`

Device pointer to array of unsigned ints, one per curve segment. This buffer is required (unlike for [OptixBuildInputTriangleArray](#)). Each index is the start of degree+1 consecutive vertices in [vertexBuffers](#), and corresponding widths in [widthBuffers](#) and normals in [normalBuffers](#). These define a single segment. Size of array is [numPrimitives](#).

### 7.6.2.5 indexStrideInBytes

`unsigned int OptixBuildInputCurveArray::indexStrideInBytes`

Stride between indices. If set to zero, indices are assumed to be tightly packed and stride is `sizeof(unsigned int)`.

### 7.6.2.6 normalBuffers

`const CUdeviceptr* OptixBuildInputCurveArray::normalBuffers`

Reserved for future use.

### 7.6.2.7 normalStrideInBytes

`unsigned int OptixBuildInputCurveArray::normalStrideInBytes`

Reserved for future use.

### 7.6.2.8 numPrimitives

`unsigned int OptixBuildInputCurveArray::numPrimitives`

Number of primitives. Each primitive is a polynomial curve segment.

### 7.6.2.9 numVertices

`unsigned int OptixBuildInputCurveArray::numVertices`

Number of vertices in each buffer in [vertexBuffers](#).

### 7.6.2.10 primitiveIndexOffset

`unsigned int OptixBuildInputCurveArray::primitiveIndexOffset`

Primitive index bias, applied in [optixGetPrimitiveIndex\(\)](#). Sum of [primitiveIndexOffset](#) and number of

primitives must not overflow 32bits.

#### 7.6.2.11 vertexBuffers

const CUdeviceptr\* OptixBuildInputCurveArray::vertexBuffers

Pointer to host array of device pointers, one per motion step. Host array size must match number of motion keys as set in [OptixMotionOptions](#) (or an array of size 1 if [OptixMotionOptions::numKeys](#) is set to 1). Each per-motion-key device pointer must point to an array of floats (the vertices of the curves).

#### 7.6.2.12 vertexStrideInBytes

unsigned int OptixBuildInputCurveArray::vertexStrideInBytes

Stride between vertices. If set to zero, vertices are assumed to be tightly packed and stride is sizeof(float3).

#### 7.6.2.13 widthBuffers

const CUdeviceptr\* OptixBuildInputCurveArray::widthBuffers

Parallel to vertexBuffers: a device pointer per motion step, each with numVertices float values, specifying the curve width (radius) corresponding to each vertex.

#### 7.6.2.14 widthStrideInBytes

unsigned int OptixBuildInputCurveArray::widthStrideInBytes

Stride between widths. If set to zero, widths are assumed to be tightly packed and stride is sizeof(float).

### 7.7 OptixBuildInputCustomPrimitiveArray Struct Reference

```
#include <optix_types.h>
```

#### Public Attributes

- const CUdeviceptr \* aabbBuffers
- unsigned int numPrimitives
- unsigned int strideInBytes
- const unsigned int \* flags
- unsigned int numSbtRecords
- CUdeviceptr sbtIndexOffsetBuffer
- unsigned int sbtIndexOffsetSizeInBytes
- unsigned int sbtIndexOffsetStrideInBytes
- unsigned int primitiveIndexOffset

#### 7.7.1 Detailed Description

Custom primitive inputs.

See also [OptixBuildInput::customPrimitiveArray](#)

#### 7.7.2 Member Data Documentation

##### 7.7.2.1 aabbBuffers

const CUdeviceptr\* OptixBuildInputCustomPrimitiveArray::aabbBuffers

Points to host array of device pointers to AABBs (type [OptixAabb](#)), one per motion step. Host array size must match number of motion keys as set in [OptixMotionOptions](#) (or an array of size 1 if [OptixMotionOptions::numKeys](#) is set to 1). Each device pointer must be a multiple of `OPTIX_AABB_BUFFER_BYTE_ALIGNMENT`.

### 7.7.2.2 flags

```
const unsigned int* OptixBuildInputCustomPrimitiveArray::flags
```

Array of flags, to specify flags per sbt record, combinations of [OptixGeometryFlags](#) describing the primitive behavior, size must match `numSbtRecords`.

### 7.7.2.3 numPrimitives

```
unsigned int OptixBuildInputCustomPrimitiveArray::numPrimitives
```

Number of primitives in each buffer (i.e., per motion step) in [OptixBuildInputCustomPrimitiveArray::aabbBuffers](#).

### 7.7.2.4 numSbtRecords

```
unsigned int OptixBuildInputCustomPrimitiveArray::numSbtRecords
```

Number of sbt records available to the sbt index offset override.

### 7.7.2.5 primitiveIndexOffset

```
unsigned int OptixBuildInputCustomPrimitiveArray::primitiveIndexOffset
```

Primitive index bias, applied in [optixGetPrimitiveIndex\(\)](#). Sum of `primitiveIndexOffset` and number of primitive must not overflow 32bits.

### 7.7.2.6 sbtIndexOffsetBuffer

```
CUdeviceptr OptixBuildInputCustomPrimitiveArray::sbtIndexOffsetBuffer
```

Device pointer to per-primitive local sbt index offset buffer. May be NULL. Every entry must be in range `[0,numSbtRecords-1]`. Size needs to be the number of primitives.

### 7.7.2.7 sbtIndexOffsetSizeInBytes

```
unsigned int OptixBuildInputCustomPrimitiveArray::sbtIndexOffsetSizeInBytes
```

Size of type of the sbt index offset. Needs to be 0, 1, 2 or 4 (8, 16 or 32 bit).

### 7.7.2.8 sbtIndexOffsetStrideInBytes

```
unsigned int OptixBuildInputCustomPrimitiveArray  
::sbtIndexOffsetStrideInBytes
```

Stride between the index offsets. If set to zero, the offsets are assumed to be tightly packed and the stride matches the size of the type (`sbtIndexOffsetSizeInBytes`).

### 7.7.2.9 strideInBytes

```
unsigned int OptixBuildInputCustomPrimitiveArray::strideInBytes
```

Stride between AABBs (per motion key). If set to zero, the aabbs are assumed to be tightly packed and the stride is assumed to be `sizeof(OptixAabb)`. If non-zero, the value must be a multiple of `OPTIX_AABB_BUFFER_BYTE_ALIGNMENT`.

## 7.8 OptixBuildInputDisplacementMicromap Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- `OptixDisplacementMicromapArrayIndexingMode` indexingMode
- `CUdeviceptr` displacementMicromapArray
- `CUdeviceptr` displacementMicromapIndexBuffer
- `CUdeviceptr` vertexDirectionsBuffer
- `CUdeviceptr` vertexBiasAndScaleBuffer
- `CUdeviceptr` triangleFlagsBuffer
- `unsigned int` displacementMicromapIndexOffset
- `unsigned int` displacementMicromapIndexStrideInBytes
- `unsigned int` displacementMicromapIndexSizeInBytes
- `OptixDisplacementMicromapDirectionFormat` vertexDirectionFormat
- `unsigned int` vertexDirectionStrideInBytes
- `OptixDisplacementMicromapBiasAndScaleFormat` vertexBiasAndScaleFormat
- `unsigned int` vertexBiasAndScaleStrideInBytes
- `unsigned int` triangleFlagsStrideInBytes
- `unsigned int` numDisplacementMicromapUsageCounts
- `const OptixDisplacementMicromapUsageCount * displacementMicromapUsageCounts`

### 7.8.1 Detailed Description

Optional displacement part of a triangle array input.

### 7.8.2 Member Data Documentation

#### 7.8.2.1 displacementMicromapArray

`CUdeviceptr` `OptixBuildInputDisplacementMicromap::displacementMicromapArray`

Address to a displacement micromap array used by this build input array. Set to NULL to disable DMs for this input.

#### 7.8.2.2 displacementMicromapIndexBuffer

`CUdeviceptr` `OptixBuildInputDisplacementMicromap::displacementMicromapIndexBuffer`

`int16` or `int32` buffer specifying which displacement micromap index to use for each triangle. Only valid if `displacementMicromapArray != NULL`.

#### 7.8.2.3 displacementMicromapIndexOffset

`unsigned int` `OptixBuildInputDisplacementMicromap::displacementMicromapIndexOffset`

Constant offset to displacement micromap indices as specified by the displacement micromap index buffer.

#### 7.8.2.4 displacementMicromapIndexSizeInBytes

`unsigned int` `OptixBuildInputDisplacementMicromap::displacementMicromapIndexSizeInBytes`

2 or 4 (16 or 32 bit)

### 7.8.2.5 displacementMicromapIndexStrideInBytes

```
unsigned int OptixBuildInputDisplacementMicromap
::displacementMicromapIndexStrideInBytes
```

Displacement micromap index buffer stride. If set to zero, indices are assumed to be tightly packed and stride is inferred from [OptixBuildInputDisplacementMicromap::displacementMicromapIndexSizeInBytes](#).

### 7.8.2.6 displacementMicromapUsageCounts

```
const OptixDisplacementMicromapUsageCount*
OptixBuildInputDisplacementMicromap::displacementMicromapUsageCounts
```

List of number of usages of displacement micromaps of format and subdivision combinations. Counts with equal format and subdivision combination (duplicates) are added together.

### 7.8.2.7 indexingMode

```
OptixDisplacementMicromapArrayIndexingMode
OptixBuildInputDisplacementMicromap::indexingMode
```

Indexing mode of triangle to displacement micromap array mapping.

### 7.8.2.8 numDisplacementMicromapUsageCounts

```
unsigned int OptixBuildInputDisplacementMicromap
::numDisplacementMicromapUsageCounts
```

Number of [OptixDisplacementMicromapUsageCount](#) entries.

### 7.8.2.9 triangleFlagsBuffer

```
CUdeviceptr OptixBuildInputDisplacementMicromap::triangleFlagsBuffer
```

Optional per-triangle flags, uint8\_t per triangle, possible values defined in enum [OptixDisplacementMicromapTriangleFlags](#).

### 7.8.2.10 triangleFlagsStrideInBytes

```
unsigned int OptixBuildInputDisplacementMicromap::triangleFlagsStrideInBytes
```

Stride in bytes for triangleFlags.

### 7.8.2.11 vertexBiasAndScaleBuffer

```
CUdeviceptr OptixBuildInputDisplacementMicromap::vertexBiasAndScaleBuffer
```

Optional per-vertex bias (offset) along displacement direction and displacement direction scale.

### 7.8.2.12 vertexBiasAndScaleFormat

```
OptixDisplacementMicromapBiasAndScaleFormat
OptixBuildInputDisplacementMicromap::vertexBiasAndScaleFormat
```

Format of vertex bias and direction scale.

### 7.8.2.13 vertexBiasAndScaleStrideInBytes

```
unsigned int OptixBuildInputDisplacementMicromap
```

`::vertexBiasAndScaleStrideInBytes`

Stride in bytes for vertex bias and direction scale entries.

#### 7.8.2.14 vertexDirectionFormat

`OptixDisplacementMicromapDirectionFormat`

`OptixBuildInputDisplacementMicromap::vertexDirectionFormat`

Format of displacement vectors.

#### 7.8.2.15 vertexDirectionsBuffer

`CUdeviceptr OptixBuildInputDisplacementMicromap::vertexDirectionsBuffer`

Per triangle-vertex displacement directions.

#### 7.8.2.16 vertexDirectionStrideInBytes

`unsigned int OptixBuildInputDisplacementMicromap  
::vertexDirectionStrideInBytes`

Stride between displacement vectors.

### 7.9 OptixBuildInputInstanceArray Struct Reference

`#include <optix_types.h>`

#### Public Attributes

- `CUdeviceptr` instances
- unsigned int `numInstances`
- unsigned int `instanceStride`

#### 7.9.1 Detailed Description

Instance and instance pointer inputs.

See also `OptixBuildInput::instanceArray`

#### 7.9.2 Member Data Documentation

##### 7.9.2.1 instances

`CUdeviceptr OptixBuildInputInstanceArray::instances`

If `OptixBuildInput::type` is `OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS` instances and aabbs should be interpreted as arrays of pointers instead of arrays of structs.

This pointer must be a multiple of `OPTIX_INSTANCE_BYTE_ALIGNMENT` if `OptixBuildInput::type` is `OPTIX_BUILD_INPUT_TYPE_INSTANCES`. The array elements must be a multiple of `OPTIX_INSTANCE_BYTE_ALIGNMENT` if `OptixBuildInput::type` is `OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS`.

##### 7.9.2.2 instanceStride

`unsigned int OptixBuildInputInstanceArray::instanceStride`

Only valid for `OPTIX_BUILD_INPUT_TYPE_INSTANCE` Defines the stride between instances. A stride of 0 indicates a tight packing, i.e., `stride = sizeof(OptixInstance)`

### 7.9.2.3 numInstances

`unsigned int OptixBuildInputInstanceArray::numInstances`

Number of elements in `OptixBuildInputInstanceArray::instances`.

## 7.10 OptixBuildInputOpacityMicromap Struct Reference

`#include <optix_types.h>`

### Public Attributes

- `OptixOpacityMicromapArrayIndexingMode indexingMode`
- `CUdeviceptr opacityMicromapArray`
- `CUdeviceptr indexBuffer`
- `unsigned int indexSizeInBytes`
- `unsigned int indexStrideInBytes`
- `unsigned int indexOffset`
- `unsigned int numMicromapUsageCounts`
- `const OptixOpacityMicromapUsageCount * micromapUsageCounts`

### 7.10.1 Member Data Documentation

#### 7.10.1.1 indexBuffer

`CUdeviceptr OptixBuildInputOpacityMicromap::indexBuffer`

int16 or int32 buffer specifying which opacity micromap index to use for each triangle. Instead of an actual index, one of the predefined indices `OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT` | `OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE` | `OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT` | `OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE` can be used to indicate that there is no opacity micromap for this particular triangle but the triangle is in a uniform state and the selected behavior is applied to the entire triangle. This buffer is required when `OptixBuildInputOpacityMicromap::indexingMode` is `OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED`. Must be zero if `OptixBuildInputOpacityMicromap::indexingMode` is `OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR` or `OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE`.

#### 7.10.1.2 indexingMode

`OptixOpacityMicromapArrayIndexingMode OptixBuildInputOpacityMicromap::indexingMode`

Indexing mode of triangle to opacity micromap array mapping.

#### 7.10.1.3 indexOffset

`unsigned int OptixBuildInputOpacityMicromap::indexOffset`

Constant offset to non-negative opacity micromap indices.

#### 7.10.1.4 indexSizeInBytes

`unsigned int OptixBuildInputOpacityMicromap::indexSizeInBytes`

0, 2 or 4 (unused, 16 or 32 bit) Must be non-zero when `OptixBuildInputOpacityMicromap::indexingMode` is `OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED`.



### 7.10.1.5 indexStrideInBytes

`unsigned int OptixBuildInputOpacityMicromap::indexStrideInBytes`

Opacity micromap index buffer stride. If set to zero, indices are assumed to be tightly packed and stride is inferred from `OptixBuildInputOpacityMicromap::indexSizeInBytes`.

### 7.10.1.6 micromapUsageCounts

`const OptixOpacityMicromapUsageCount* OptixBuildInputOpacityMicromap::micromapUsageCounts`

List of number of usages of opacity micromaps of format and subdivision combinations. Counts with equal format and subdivision combination (duplicates) are added together.

### 7.10.1.7 numMicromapUsageCounts

`unsigned int OptixBuildInputOpacityMicromap::numMicromapUsageCounts`

Number of `OptixOpacityMicromapUsageCount`.

### 7.10.1.8 opacityMicromapArray

`CUdeviceptr OptixBuildInputOpacityMicromap::opacityMicromapArray`

Device pointer to a opacity micromap array used by this build input array. This buffer is required when `OptixBuildInputOpacityMicromap::indexingMode` is `OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR` or `OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED`. Must be zero if `OptixBuildInputOpacityMicromap::indexingMode` is `OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE`.

## 7.11 OptixBuildInputSphereArray Struct Reference

`#include <optix_types.h>`

### Public Attributes

- `const CUdeviceptr * vertexBuffers`
- `unsigned int vertexStrideInBytes`
- `unsigned int numVertices`
- `const CUdeviceptr * radiusBuffers`
- `unsigned int radiusStrideInBytes`
- `int singleRadius`
- `const unsigned int * flags`
- `unsigned int numSbtRecords`
- `CUdeviceptr sbtIndexOffsetBuffer`
- `unsigned int sbtIndexOffsetSizeInBytes`
- `unsigned int sbtIndexOffsetStrideInBytes`
- `unsigned int primitiveIndexOffset`

### 7.11.1 Detailed Description

Sphere inputs.

A sphere is defined by a center point and a radius. Each center point is represented by a vertex in the vertex buffer. There is either a single radius for all spheres, or the radii are represented by entries in the radius buffer.

The vertex buffers and radius buffers point to a host array of device pointers, one per motion step. Host array size must match the number of motion keys as set in [OptixMotionOptions](#) (or an array of size 1 if [OptixMotionOptions::numKeys](#) is set to 0 or 1). Each per motion key device pointer must point to an array of vertices corresponding to the center points of the spheres, or an array of 1 or N radii. Format OPTIX\_VERTEX\_FORMAT\_FLOAT3 is used for vertices, OPTIX\_VERTEX\_FORMAT\_FLOAT for radii.

See also [OptixBuildInput::sphereArray](#)

## 7.11.2 Member Data Documentation

### 7.11.2.1 flags

`const unsigned int* OptixBuildInputSphereArray::flags`

Array of flags, to specify flags per sbt record, combinations of [OptixGeometryFlags](#) describing the primitive behavior, size must match `numSbtRecords`.

### 7.11.2.2 numSbtRecords

`unsigned int OptixBuildInputSphereArray::numSbtRecords`

Number of sbt records available to the sbt index offset override.

### 7.11.2.3 numVertices

`unsigned int OptixBuildInputSphereArray::numVertices`

Number of vertices in each buffer in `vertexBuffers`.

### 7.11.2.4 primitiveIndexOffset

`unsigned int OptixBuildInputSphereArray::primitiveIndexOffset`

Primitive index bias, applied in [optixGetPrimitiveIndex\(\)](#). Sum of `primitiveIndexOffset` and number of primitives must not overflow 32bits.

### 7.11.2.5 radiusBuffers

`const CUdeviceptr* OptixBuildInputSphereArray::radiusBuffers`

Parallel to `vertexBuffers`: a device pointer per motion step, each with `numRadii` float values, specifying the sphere radius corresponding to each vertex.

### 7.11.2.6 radiusStrideInBytes

`unsigned int OptixBuildInputSphereArray::radiusStrideInBytes`

Stride between radii. If set to zero, widths are assumed to be tightly packed and stride is `sizeof(float)`.

### 7.11.2.7 sbtIndexOffsetBuffer

`CUdeviceptr OptixBuildInputSphereArray::sbtIndexOffsetBuffer`

Device pointer to per-primitive local sbt index offset buffer. May be NULL. Every entry must be in range `[0,numSbtRecords-1]`. Size needs to be the number of primitives.

### 7.11.2.8 sbtIndexOffsetSizeInBytes

`unsigned int OptixBuildInputSphereArray::sbtIndexOffsetSizeInBytes`

Size of type of the sbt index offset. Needs to be 0, 1, 2 or 4 (8, 16 or 32 bit).

### 7.11.2.9 sbtIndexOffsetStrideInBytes

`unsigned int OptixBuildInputSphereArray::sbtIndexOffsetStrideInBytes`

Stride between the sbt index offsets. If set to zero, the offsets are assumed to be tightly packed and the stride matches the size of the type (`sbtIndexOffsetSizeInBytes`).

### 7.11.2.10 singleRadius

`int OptixBuildInputSphereArray::singleRadius`

Boolean value indicating whether a single radius per radius buffer is used, or the number of radii in `radiusBuffers` equals `numVertices`.

### 7.11.2.11 vertexBuffers

`const CUdeviceptr* OptixBuildInputSphereArray::vertexBuffers`

Pointer to host array of device pointers, one per motion step. Host array size must match number of motion keys as set in `OptixMotionOptions` (or an array of size 1 if `OptixMotionOptions::numKeys` is set to 1). Each per-motion-key device pointer must point to an array of floats (the center points of the spheres).

### 7.11.2.12 vertexStrideInBytes

`unsigned int OptixBuildInputSphereArray::vertexStrideInBytes`

Stride between vertices. If set to zero, vertices are assumed to be tightly packed and stride is `sizeof(float3)`.

## 7.12 OptixBuildInputTriangleArray Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- `const CUdeviceptr * vertexBuffers`
- `unsigned int numVertices`
- `OptixVertexFormat vertexFormat`
- `unsigned int vertexStrideInBytes`
- `CUdeviceptr indexBuffer`
- `unsigned int numIndexTriplets`
- `OptixIndicesFormat indexFormat`
- `unsigned int indexStrideInBytes`
- `CUdeviceptr preTransform`
- `const unsigned int * flags`
- `unsigned int numSbtRecords`
- `CUdeviceptr sbtIndexOffsetBuffer`
- `unsigned int sbtIndexOffsetSizeInBytes`
- `unsigned int sbtIndexOffsetStrideInBytes`
- `unsigned int primitiveIndexOffset`
- `OptixTransformFormat transformFormat`
- `OptixBuildInputOpacityMicromap opacityMicromap`
- `OptixBuildInputDisplacementMicromap displacementMicromap`

### 7.12.1 Detailed Description

Triangle inputs.

See also [OptixBuildInput::triangleArray](#)

### 7.12.2 Member Data Documentation

#### 7.12.2.1 displacementMicromap

[OptixBuildInputDisplacementMicromap](#) [OptixBuildInputTriangleArray::displacementMicromap](#)

Optional displacement micromap inputs.

#### 7.12.2.2 flags

`const unsigned int* OptixBuildInputTriangleArray::flags`

Array of flags, to specify flags per sbt record, combinations of [OptixGeometryFlags](#) describing the primitive behavior, size must match `numSbtRecords`.

#### 7.12.2.3 indexBuffer

[CUdeviceptr](#) [OptixBuildInputTriangleArray::indexBuffer](#)

Optional pointer to array of 16 or 32-bit int triplets, one triplet per triangle. The minimum alignment must match the natural alignment of the type as specified in the `indexFormat`, i.e., for `OPTIX_INDICES_FORMAT_UNSIGNED_INT3` 4-byte and for `OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3` a 2-byte alignment.

#### 7.12.2.4 indexFormat

[OptixIndicesFormat](#) [OptixBuildInputTriangleArray::indexFormat](#)

See also [OptixIndicesFormat](#)

#### 7.12.2.5 indexStrideInBytes

`unsigned int OptixBuildInputTriangleArray::indexStrideInBytes`

Stride between triplets of indices. If set to zero, indices are assumed to be tightly packed and stride is inferred from `indexFormat`.

#### 7.12.2.6 numIndexTriplets

`unsigned int OptixBuildInputTriangleArray::numIndexTriplets`

Size of array in [OptixBuildInputTriangleArray::indexBuffer](#). For build, needs to be zero if `indexBuffer` is `nullptr`.

#### 7.12.2.7 numSbtRecords

`unsigned int OptixBuildInputTriangleArray::numSbtRecords`

Number of sbt records available to the sbt index offset override.

#### 7.12.2.8 numVertices

`unsigned int OptixBuildInputTriangleArray::numVertices`

Number of vertices in each of buffer in [OptixBuildInputTriangleArray::vertexBuffers](#).

#### 7.12.2.9 opacityMicromap

[OptixBuildInputOpacityMicromap](#) [OptixBuildInputTriangleArray::opacityMicromap](#)

Optional opacity micromap inputs.

#### 7.12.2.10 preTransform

[CUdeviceptr](#) [OptixBuildInputTriangleArray::preTransform](#)

Optional pointer to array of floats representing a 3x4 row major affine transformation matrix. This pointer must be a multiple of `OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT`.

#### 7.12.2.11 primitiveIndexOffset

`unsigned int` [OptixBuildInputTriangleArray::primitiveIndexOffset](#)

Primitive index bias, applied in [optixGetPrimitiveIndex\(\)](#). Sum of `primitiveIndexOffset` and number of triangles must not overflow 32bits.

#### 7.12.2.12 sbtIndexOffsetBuffer

[CUdeviceptr](#) [OptixBuildInputTriangleArray::sbtIndexOffsetBuffer](#)

Device pointer to per-primitive local sbt index offset buffer. May be NULL. Every entry must be in range `[0,numSbtRecords-1]`. Size needs to be the number of primitives.

#### 7.12.2.13 sbtIndexOffsetSizeInBytes

`unsigned int` [OptixBuildInputTriangleArray::sbtIndexOffsetSizeInBytes](#)

Size of type of the sbt index offset. Needs to be 0, 1, 2 or 4 (8, 16 or 32 bit).

#### 7.12.2.14 sbtIndexOffsetStrideInBytes

`unsigned int` [OptixBuildInputTriangleArray::sbtIndexOffsetStrideInBytes](#)

Stride between the index offsets. If set to zero, the offsets are assumed to be tightly packed and the stride matches the size of the type (`sbtIndexOffsetSizeInBytes`).

#### 7.12.2.15 transformFormat

[OptixTransformFormat](#) [OptixBuildInputTriangleArray::transformFormat](#)

See also [OptixTransformFormat](#)

#### 7.12.2.16 vertexBuffers

`const CUdeviceptr*` [OptixBuildInputTriangleArray::vertexBuffers](#)

Points to host array of device pointers, one per motion step. Host array size must match the number of motion keys as set in [OptixMotionOptions](#) (or an array of size 1 if [OptixMotionOptions::numKeys](#) is set to 0 or 1). Each per motion key device pointer must point to an array of vertices of the triangles in the format as described by `vertexFormat`. The minimum alignment must match the natural alignment of the type as specified in the `vertexFormat`, i.e., for `OPTIX_VERTEX_FORMAT_FLOATX` 4-byte, for all others a 2-byte alignment. However, an 16-byte stride (and buffer alignment) is recommended for vertices of format `OPTIX_VERTEX_FORMAT_FLOAT3` for GAS build performance.

### 7.12.2.17 vertexFormat

[OptixVertexFormat](#) `OptixBuildInputTriangleArray::vertexFormat`

See also [OptixVertexFormat](#)

### 7.12.2.18 vertexStrideInBytes

`unsigned int OptixBuildInputTriangleArray::vertexStrideInBytes`

Stride between vertices. If set to zero, vertices are assumed to be tightly packed and stride is inferred from `vertexFormat`.

## 7.13 OptixBuiltinISOOptions Struct Reference

`#include <optix_types.h>`

### Public Attributes

- [OptixPrimitiveType](#) `builtinISModuleType`
- `int` `usesMotionBlur`
- `unsigned int` `buildFlags`
- `unsigned int` `curveEndcapFlags`

### 7.13.1 Detailed Description

Specifies the options for retrieving an intersection program for a built-in primitive type. The primitive type must not be `OPTIX_PRIMITIVE_TYPE_CUSTOM`.

See also [optixBuiltinISModuleGet\(\)](#)

### 7.13.2 Member Data Documentation

#### 7.13.2.1 buildFlags

`unsigned int OptixBuiltinISOOptions::buildFlags`

Build flags, see [OptixBuildFlags](#).

#### 7.13.2.2 builtinISModuleType

[OptixPrimitiveType](#) `OptixBuiltinISOOptions::builtinISModuleType`

#### 7.13.2.3 curveEndcapFlags

`unsigned int OptixBuiltinISOOptions::curveEndcapFlags`

End cap properties of curves, see [OptixCurveEndcapFlags](#), 0 for non-curve types.

#### 7.13.2.4 usesMotionBlur

`int OptixBuiltinISOOptions::usesMotionBlur`

Boolean value indicating whether vertex motion blur is used (but not motion transform blur).

## 7.14 OptixDenoiserGuideLayer Struct Reference

`#include <optix_types.h>`

## Public Attributes

- [OptixImage2D](#) `albedo`
- [OptixImage2D](#) `normal`
- [OptixImage2D](#) `flow`
- [OptixImage2D](#) `previousOutputInternalGuideLayer`
- [OptixImage2D](#) `outputInternalGuideLayer`
- [OptixImage2D](#) `flowTrustworthiness`

### 7.14.1 Detailed Description

Guide layer for the denoiser.

See also [optixDenoiserInvoke\(\)](#)

### 7.14.2 Member Data Documentation

#### 7.14.2.1 `albedo`

[OptixImage2D](#) `OptixDenoiserGuideLayer::albedo`

#### 7.14.2.2 `flow`

[OptixImage2D](#) `OptixDenoiserGuideLayer::flow`

#### 7.14.2.3 `flowTrustworthiness`

[OptixImage2D](#) `OptixDenoiserGuideLayer::flowTrustworthiness`

#### 7.14.2.4 `normal`

[OptixImage2D](#) `OptixDenoiserGuideLayer::normal`

#### 7.14.2.5 `outputInternalGuideLayer`

[OptixImage2D](#) `OptixDenoiserGuideLayer::outputInternalGuideLayer`

#### 7.14.2.6 `previousOutputInternalGuideLayer`

[OptixImage2D](#) `OptixDenoiserGuideLayer::previousOutputInternalGuideLayer`

## 7.15 OptixDenoiserLayer Struct Reference

```
#include <optix_types.h>
```

## Public Attributes

- [OptixImage2D](#) `input`
- [OptixImage2D](#) `previousOutput`
- [OptixImage2D](#) `output`
- [OptixDenoiserAOVType](#) `type`

### 7.15.1 Detailed Description

Input/Output layers for the denoiser.

See also [optixDenoiserInvoke\(\)](#)

## 7.15.2 Member Data Documentation

### 7.15.2.1 input

[OptixImage2D](#) [OptixDenoiserLayer::input](#)

### 7.15.2.2 output

[OptixImage2D](#) [OptixDenoiserLayer::output](#)

### 7.15.2.3 previousOutput

[OptixImage2D](#) [OptixDenoiserLayer::previousOutput](#)

### 7.15.2.4 type

[OptixDenoiserAOVType](#) [OptixDenoiserLayer::type](#)

## 7.16 OptixDenoiserOptions Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- unsigned int [guideAlbedo](#)
- unsigned int [guideNormal](#)

### 7.16.1 Detailed Description

Options used by the denoiser.

See also [optixDenoiserCreate\(\)](#)

## 7.16.2 Member Data Documentation

### 7.16.2.1 guideAlbedo

unsigned int [OptixDenoiserOptions::guideAlbedo](#)

### 7.16.2.2 guideNormal

unsigned int [OptixDenoiserOptions::guideNormal](#)

## 7.17 OptixDenoiserParams Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- [OptixDenoiserAlphaMode](#) [denoiseAlpha](#)
- [CUdeviceptr](#) [hdrIntensity](#)
- float [blendFactor](#)
- [CUdeviceptr](#) [hdrAverageColor](#)
- unsigned int [temporalModeUsePreviousLayers](#)



## 7.17.1 Member Data Documentation

### 7.17.1.1 blendFactor

`float OptixDenoiserParams::blendFactor`

blend factor. If set to 0 the output is 100% of the denoised input. If set to 1, the output is 100% of the unmodified input. Values between 0 and 1 will linearly interpolate between the denoised and unmodified input.

### 7.17.1.2 denoiseAlpha

`OptixDenoiserAlphaMode OptixDenoiserParams::denoiseAlpha`

alpha denoise mode

### 7.17.1.3 hdrAverageColor

`CUdeviceptr OptixDenoiserParams::hdrAverageColor`

this parameter is used when the `OPTIX_DENOISER_MODEL_KIND_AOV` model kind is set. average log color of input image, separate for RGB channels (default null pointer). points to three floats. with the default (null pointer) denoised results will not be optimal.

### 7.17.1.4 hdrIntensity

`CUdeviceptr OptixDenoiserParams::hdrIntensity`

average log intensity of input image (default null pointer). points to a single float. with the default (null pointer) denoised results will not be optimal for very dark or bright input images.

### 7.17.1.5 temporalModeUsePreviousLayers

`unsigned int OptixDenoiserParams::temporalModeUsePreviousLayers`

In temporal modes this parameter must be set to 1 if previous layers (e.g. `previousOutputInternalGuideLayer`) contain valid data. This is the case in the second and subsequent frames of a sequence (for example after a change of camera angle). In the first frame of such a sequence this parameter must be set to 0.

## 7.18 OptixDenoiserSizes Struct Reference

`#include <optix_types.h>`

### Public Attributes

- `size_t stateSizeInBytes`
- `size_t withOverlapScratchSizeInBytes`
- `size_t withoutOverlapScratchSizeInBytes`
- `unsigned int overlapWindowSizeInPixels`
- `size_t computeAverageColorSizeInBytes`
- `size_t computeIntensitySizeInBytes`
- `size_t internalGuideLayerPixelSizeInBytes`

### 7.18.1 Detailed Description

Various sizes related to the denoiser.

See also `optixDenoiserComputeMemoryResources()`

## 7.18.2 Member Data Documentation

### 7.18.2.1 computeAverageColorSizeInBytes

`size_t OptixDenoiserSizes::computeAverageColorSizeInBytes`

Size of scratch memory passed to `optixDenoiserComputeAverageColor`. The size is independent of the tile/image resolution.

### 7.18.2.2 computeIntensitySizeInBytes

`size_t OptixDenoiserSizes::computeIntensitySizeInBytes`

Size of scratch memory passed to `optixDenoiserComputeIntensity`. The size is independent of the tile/image resolution.

### 7.18.2.3 internalGuideLayerPixelSizeInBytes

`size_t OptixDenoiserSizes::internalGuideLayerPixelSizeInBytes`

Number of bytes for each pixel in internal guide layers.

### 7.18.2.4 overlapWindowSizeInPixels

`unsigned int OptixDenoiserSizes::overlapWindowSizeInPixels`

Overlap on all four tile sides.

### 7.18.2.5 stateSizeInBytes

`size_t OptixDenoiserSizes::stateSizeInBytes`

Size of state memory passed to `optixDenoiserSetup`, `optixDenoiserInvoke`.

### 7.18.2.6 withoutOverlapScratchSizeInBytes

`size_t OptixDenoiserSizes::withoutOverlapScratchSizeInBytes`

Size of scratch memory passed to `optixDenoiserSetup`, `optixDenoiserInvoke`. No overlap added.

### 7.18.2.7 withOverlapScratchSizeInBytes

`size_t OptixDenoiserSizes::withOverlapScratchSizeInBytes`

Size of scratch memory passed to `optixDenoiserSetup`, `optixDenoiserInvoke`. Overlap added to dimensions passed to `optixDenoiserComputeMemoryResources`.

## 7.19 OptixDeviceContextOptions Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- `OptixLogCallback logCallbackFunction`
- `void * logCallbackData`
- `int logCallbackLevel`
- `OptixDeviceContextValidationMode validationMode`

### 7.19.1 Detailed Description

Parameters used for [optixDeviceContextCreate\(\)](#)

See also [optixDeviceContextCreate\(\)](#)

### 7.19.2 Member Data Documentation

#### 7.19.2.1 logCallbackData

`void* OptixDeviceContextOptions::logCallbackData`

Pointer stored and passed to `logCallbackFunction` when a message is generated.

#### 7.19.2.2 logCallbackFunction

`OptixLogCallback OptixDeviceContextOptions::logCallbackFunction`

Function pointer used when OptiX wishes to generate messages.

#### 7.19.2.3 logCallbackLevel

`int OptixDeviceContextOptions::logCallbackLevel`

Maximum callback level to generate message for (see [OptixLogCallback](#))

#### 7.19.2.4 validationMode

`OptixDeviceContextValidationMode OptixDeviceContextOptions::validationMode`

Validation mode of context.

## 7.20 OptixDisplacementMicromapArrayBuildInput Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- [OptixDisplacementMicromapFlags](#) flags
- `CUdeviceptr` displacementValuesBuffer
- `CUdeviceptr` perDisplacementMicromapDescBuffer
- unsigned int perDisplacementMicromapDescStrideInBytes
- unsigned int numDisplacementMicromapHistogramEntries
- const [OptixDisplacementMicromapHistogramEntry](#) \* displacementMicromapHistogramEntries

### 7.20.1 Detailed Description

Inputs to displacement micromaps array construction.

### 7.20.2 Member Data Documentation

#### 7.20.2.1 displacementMicromapHistogramEntries

`const OptixDisplacementMicromapHistogramEntry*`  
`OptixDisplacementMicromapArrayBuildInput`  
`::displacementMicromapHistogramEntries`

Histogram over DMMs for input format and subdivision combinations. Counts of histogram bins with equal format and subdivision combinations are added together.

### 7.20.2.2 displacementValuesBuffer

`CUdeviceptr` `OptixDisplacementMicromapArrayBuildInput`  
`::displacementValuesBuffer`

128 byte aligned pointer for displacement micromap raw input data.

### 7.20.2.3 flags

`OptixDisplacementMicromapFlags` `OptixDisplacementMicromapArrayBuildInput`  
`::flags`

Flags that apply to all displacement micromaps in array.

### 7.20.2.4 numDisplacementMicromapHistogramEntries

`unsigned int` `OptixDisplacementMicromapArrayBuildInput`  
`::numDisplacementMicromapHistogramEntries`

Number of `OptixDisplacementMicromapHistogramEntry` entries.

### 7.20.2.5 perDisplacementMicromapDescBuffer

`CUdeviceptr` `OptixDisplacementMicromapArrayBuildInput`  
`::perDisplacementMicromapDescBuffer`

Descriptors for interpreting raw input data, one `OptixDisplacementMicromapDesc` entry required per displacement micromap. This device pointer must be a multiple of `OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT`.

### 7.20.2.6 perDisplacementMicromapDescStrideInBytes

`unsigned int` `OptixDisplacementMicromapArrayBuildInput`  
`::perDisplacementMicromapDescStrideInBytes`

Stride between `OptixDisplacementMicromapDesc` in `perDisplacementMicromapDescBuffer`. If set to zero, the displacement micromap descriptors are assumed to be tightly packed and the stride is assumed to be `sizeof(OptixDisplacementMicromapDesc)`. This stride must be a multiple of `OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT`.

## 7.21 OptixDisplacementMicromapDesc Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- `unsigned int` `byteOffset`
- `unsigned short` `subdivisionLevel`
- `unsigned short` `format`

### 7.21.1 Member Data Documentation

#### 7.21.1.1 byteOffset

`unsigned int` `OptixDisplacementMicromapDesc::byteOffset`

Block is located at `displacementValuesBuffer + byteOffset`.

### 7.21.1.2 format

unsigned short OptixDisplacementMicromapDesc::format

Format (OptixDisplacementMicromapFormat)

### 7.21.1.3 subdivisionLevel

unsigned short OptixDisplacementMicromapDesc::subdivisionLevel

Number of micro-triangles is  $4^{\text{level}}$ . Valid levels are [0, 5].

## 7.22 OptixDisplacementMicromapHistogramEntry Struct Reference

#include <optix\_types.h>

### Public Attributes

- unsigned int [count](#)
- unsigned int [subdivisionLevel](#)
- [OptixDisplacementMicromapFormat](#) format

### 7.22.1 Detailed Description

Displacement micromap histogram entry. Specifies how many displacement micromaps of a specific type are input to the displacement micromap array build. Note that while this is similar to [OptixDisplacementMicromapUsageCount](#), the histogram entry specifies how many displacement micromaps of a specific type are combined into a displacement micromap array.

### 7.22.2 Member Data Documentation

#### 7.22.2.1 count

unsigned int OptixDisplacementMicromapHistogramEntry::count

Number of displacement micromaps with the format and subdivision level that are input to the displacement micromap array build.

#### 7.22.2.2 format

[OptixDisplacementMicromapFormat](#) OptixDisplacementMicromapHistogramEntry::format

Displacement micromap format.

#### 7.22.2.3 subdivisionLevel

unsigned int OptixDisplacementMicromapHistogramEntry::subdivisionLevel

Number of micro-triangles is  $4^{\text{level}}$ . Valid levels are [0, 5].

## 7.23 OptixDisplacementMicromapUsageCount Struct Reference

#include <optix\_types.h>

### Public Attributes

- unsigned int [count](#)
- unsigned int [subdivisionLevel](#)
- [OptixDisplacementMicromapFormat](#) format

### 7.23.1 Detailed Description

Displacement micromap usage count for acceleration structure builds. Specifies how many displacement micromaps of a specific type are referenced by triangles when building the AS. Note that while this is similar to [OptixDisplacementMicromapHistogramEntry](#), the usage count specifies how many displacement micromaps of a specific type are referenced by triangles in the AS.

### 7.23.2 Member Data Documentation

#### 7.23.2.1 count

`unsigned int OptixDisplacementMicromapUsageCount::count`

Number of displacement micromaps with this format and subdivision level referenced by triangles in the corresponding triangle build input at AS build time.

#### 7.23.2.2 format

`OptixDisplacementMicromapFormat OptixDisplacementMicromapUsageCount::format`

Displacement micromaps format.

#### 7.23.2.3 subdivisionLevel

`unsigned int OptixDisplacementMicromapUsageCount::subdivisionLevel`

Number of micro-triangles is  $4^{\text{level}}$ . Valid levels are [0, 5].

## 7.24 OptixFunctionTable Struct Reference

`#include <optix_function_table.h>`

### Public Attributes

#### Error handling

- `const char *(* optixGetErrorName)(OptixResult result)`
- `const char *(* optixGetErrorString)(OptixResult result)`

#### Device context

- `OptixResult(* optixDeviceContextCreate)(CUcontext fromContext, const OptixDeviceContextOptions *options, OptixDeviceContext *context)`
- `OptixResult(* optixDeviceContextDestroy)(OptixDeviceContext context)`
- `OptixResult(* optixDeviceContextGetProperty)(OptixDeviceContext context, OptixDeviceProperty property, void *value, size_t sizeInBytes)`
- `OptixResult(* optixDeviceContextSetLogCallback)(OptixDeviceContext context, OptixLogCallback callbackFunction, void *callbackData, unsigned int callbackLevel)`
- `OptixResult(* optixDeviceContextSetCacheEnabled)(OptixDeviceContext context, int enabled)`
- `OptixResult(* optixDeviceContextSetCacheLocation)(OptixDeviceContext context, const char *location)`
- `OptixResult(* optixDeviceContextSetCacheDatabaseSizes)(OptixDeviceContext context, size_t lowWaterMark, size_t highWaterMark)`
- `OptixResult(* optixDeviceContextGetCacheEnabled)(OptixDeviceContext context, int *enabled)`
- `OptixResult(* optixDeviceContextGetCacheLocation)(OptixDeviceContext context, char *location, size_t locationSize)`
- `OptixResult(* optixDeviceContextGetCacheDatabaseSizes)(OptixDeviceContext context, size_t *lowWaterMark, size_t *highWaterMark)`

## Modules

- `OptixResult(* optixModuleCreate)(OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module)`
- `OptixResult(* optixModuleCreateWithTasks)(OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module, OptixTask *firstTask)`
- `OptixResult(* optixModuleGetCompilationState)(OptixModule module, OptixModuleCompileState *state)`
- `OptixResult(* optixModuleDestroy)(OptixModule module)`
- `OptixResult(* optixBuiltinISModuleGet)(OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixBuiltinISOptions *builtinISOptions, OptixModule *builtinModule)`

## Tasks

- `OptixResult(* optixTaskExecute)(OptixTask task, OptixTask *additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int *numAdditionalTasksCreated)`

## Program groups

- `OptixResult(* optixProgramGroupCreate)(OptixDeviceContext context, const OptixProgramGroupDesc *programDescriptions, unsigned int numProgramGroups, const OptixProgramGroupOptions *options, char *logString, size_t *logStringSize, OptixProgramGroup *programGroups)`
- `OptixResult(* optixProgramGroupDestroy)(OptixProgramGroup programGroup)`
- `OptixResult(* optixProgramGroupGetStackSize)(OptixProgramGroup programGroup, OptixStackSizes *stackSizes, OptixPipeline pipeline)`

## Pipeline

- `OptixResult(* optixPipelineCreate)(OptixDeviceContext context, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixPipelineLinkOptions *pipelineLinkOptions, const OptixProgramGroup *programGroups, unsigned int numProgramGroups, char *logString, size_t *logStringSize, OptixPipeline *pipeline)`
- `OptixResult(* optixPipelineDestroy)(OptixPipeline pipeline)`
- `OptixResult(* optixPipelineSetStackSize)(OptixPipeline pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)`

## Acceleration structures

- `OptixResult(* optixAccelComputeMemoryUsage)(OptixDeviceContext context, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, OptixAccelBufferSizes *bufferSizes)`
- `OptixResult(* optixAccelBuild)(OptixDeviceContext context, CUstream stream, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, CUdeviceptr tempBuffer, size_t tempBufferSizeInBytes, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle, const OptixAccelEmitDesc *emittedProperties, unsigned int numEmittedProperties)`
- `OptixResult(* optixAccelGetRelocationInfo)(OptixDeviceContext context, OptixTraversableHandle handle, OptixRelocationInfo *info)`
- `OptixResult(* optixCheckRelocationCompatibility)(OptixDeviceContext context, const OptixRelocationInfo *info, int *compatible)`



- `OptixResult(* optixAccelRelocate)(OptixDeviceContext context, CUstream stream, const OptixRelocationInfo *info, const OptixRelocateInput *relocateInputs, size_t numRelocateInputs, CUdeviceptr targetAccel, size_t targetAccelSizeInBytes, OptixTraversableHandle *targetHandle)`
- `OptixResult(* optixAccelCompact)(OptixDeviceContext context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle)`
- `OptixResult(* optixAccelEmitProperty)(OptixDeviceContext context, CUstream stream, OptixTraversableHandle handle, const OptixAccelEmitDesc *emittedProperty)`
- `OptixResult(* optixConvertPointerToTraversableHandle)(OptixDeviceContext onDevice, CUdeviceptr pointer, OptixTraversableType traversableType, OptixTraversableHandle *traversableHandle)`
- `OptixResult(* optixOpacityMicromapArrayComputeMemoryUsage)(OptixDeviceContext context, const OptixOpacityMicromapArrayBuildInput *buildInput, OptixMicromapBufferSizes *bufferSizes)`
- `OptixResult(* optixOpacityMicromapArrayBuild)(OptixDeviceContext context, CUstream stream, const OptixOpacityMicromapArrayBuildInput *buildInput, const OptixMicromapBuffers *buffers)`
- `OptixResult(* optixOpacityMicromapArrayGetRelocationInfo)(OptixDeviceContext context, CUdeviceptr opacityMicromapArray, OptixRelocationInfo *info)`
- `OptixResult(* optixOpacityMicromapArrayRelocate)(OptixDeviceContext context, CUstream stream, const OptixRelocationInfo *info, CUdeviceptr targetOpacityMicromapArray, size_t targetOpacityMicromapArraySizeInBytes)`
- `OptixResult(* optixDisplacementMicromapArrayComputeMemoryUsage)(OptixDeviceContext context, const OptixDisplacementMicromapArrayBuildInput *buildInput, OptixMicromapBufferSizes *bufferSizes)`
- `OptixResult(* optixDisplacementMicromapArrayBuild)(OptixDeviceContext context, CUstream stream, const OptixDisplacementMicromapArrayBuildInput *buildInput, const OptixMicromapBuffers *buffers)`

#### Launch

- `OptixResult(* optixSbtRecordPackHeader)(OptixProgramGroup programGroup, void *sbtRecordHeaderHostPointer)`
- `OptixResult(* optixLaunch)(OptixPipeline pipeline, CUstream stream, CUdeviceptr pipelineParams, size_t pipelineParamsSize, const OptixShaderBindingTable *sbt, unsigned int width, unsigned int height, unsigned int depth)`

#### Denoiser

- `OptixResult(* optixDenoiserCreate)(OptixDeviceContext context, OptixDenoiserModelKind modelKind, const OptixDenoiserOptions *options, OptixDenoiser *returnHandle)`
- `OptixResult(* optixDenoiserDestroy)(OptixDenoiser handle)`
- `OptixResult(* optixDenoiserComputeMemoryResources)(const OptixDenoiser handle, unsigned int maximumInputWidth, unsigned int maximumInputHeight, OptixDenoiserSizes *returnSizes)`
- `OptixResult(* optixDenoiserSetup)(OptixDenoiser denoiser, CUstream stream, unsigned int inputWidth, unsigned int inputHeight, CUdeviceptr state, size_t stateSizeInBytes, CUdeviceptr scratch, size_t scratchSizeInBytes)`
- `OptixResult(* optixDenoiserInvoke)(OptixDenoiser denoiser, CUstream stream, const OptixDenoiserParams *params, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, const OptixDenoiserGuideLayer *guideLayer, const OptixDenoiserLayer *layers, unsigned int numLayers, unsigned int inputOffsetX, unsigned int inputOffsetY, CUdeviceptr scratch, size_t scratchSizeInBytes)`



- `OptixResult(* optixDenoiserComputeIntensity)(OptixDenoiser handle, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputIntensity, CUdeviceptr scratch, size_t scratchSizeInBytes)`
- `OptixResult(* optixDenoiserComputeAverageColor)(OptixDenoiser handle, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputAverageColor, CUdeviceptr scratch, size_t scratchSizeInBytes)`
- `OptixResult(* optixDenoiserCreateWithUserModel)(OptixDeviceContext context, const void *data, size_t dataSizeInBytes, OptixDenoiser *returnHandle)`

### 7.24.1 Detailed Description

The function table containing all API functions.

See `optixInit()` and `optixInitWithHandle()`.

### 7.24.2 Member Data Documentation

#### 7.24.2.1 optixAccelBuild

`OptixResult(* OptixFunctionTable::optixAccelBuild) (OptixDeviceContext context, CUstream stream, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, CUdeviceptr tempBuffer, size_t tempBufferSizeInBytes, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle, const OptixAccelEmitDesc *emittedProperties, unsigned int numEmittedProperties)`

See `optixAccelBuild()`.

#### 7.24.2.2 optixAccelCompact

`OptixResult(* OptixFunctionTable::optixAccelCompact) (OptixDeviceContext context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle)`

See `optixAccelCompact()`.

#### 7.24.2.3 optixAccelComputeMemoryUsage

`OptixResult(* OptixFunctionTable::optixAccelComputeMemoryUsage) (OptixDeviceContext context, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, OptixAccelBufferSizes *bufferSizes)`

See `optixAccelComputeMemoryUsage()`.

#### 7.24.2.4 optixAccelEmitProperty

`OptixResult(* OptixFunctionTable::optixAccelEmitProperty) (OptixDeviceContext context, CUstream stream, OptixTraversableHandle handle, const OptixAccelEmitDesc *emittedProperty)`

See `optixAccelComputeMemoryUsage()`.

#### 7.24.2.5 optixAccelGetRelocationInfo

`OptixResult(* OptixFunctionTable::optixAccelGetRelocationInfo) (OptixDeviceContext context, OptixTraversableHandle handle,`

`OptixRelocationInfo *info)`

See `optixAccelGetRelocationInfo()`.

#### 7.24.2.6 `optixAccelRelocate`

`OptixResult(* OptixFunctionTable::optixAccelRelocate) (OptixDeviceContext context, CUstream stream, const OptixRelocationInfo *info, const OptixRelocateInput *relocateInputs, size_t numRelocateInputs, CUdeviceptr targetAccel, size_t targetAccelSizeInBytes, OptixTraversableHandle *targetHandle)`

See `optixAccelRelocate()`.

#### 7.24.2.7 `optixBuiltinISModuleGet`

`OptixResult(* OptixFunctionTable::optixBuiltinISModuleGet) (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixBuiltinISOptions *builtinISOptions, OptixModule *builtinModule)`

See `optixBuiltinISModuleGet()`.

#### 7.24.2.8 `optixCheckRelocationCompatibility`

`OptixResult(* OptixFunctionTable::optixCheckRelocationCompatibility) (OptixDeviceContext context, const OptixRelocationInfo *info, int *compatible)`

See `optixCheckRelocationCompatibility()`.

#### 7.24.2.9 `optixConvertPointerToTraversableHandle`

`OptixResult(* OptixFunctionTable::optixConvertPointerToTraversableHandle) (OptixDeviceContext onDevice, CUdeviceptr pointer, OptixTraversableType traversableType, OptixTraversableHandle *traversableHandle)`

See `optixConvertPointerToTraversableHandle()`.

#### 7.24.2.10 `optixDenoiserComputeAverageColor`

`OptixResult(* OptixFunctionTable::optixDenoiserComputeAverageColor) (OptixDenoiser handle, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputAverageColor, CUdeviceptr scratch, size_t scratchSizeInBytes)`

See `optixDenoiserComputeAverageColor()`.

#### 7.24.2.11 `optixDenoiserComputeIntensity`

`OptixResult(* OptixFunctionTable::optixDenoiserComputeIntensity) (OptixDenoiser handle, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputIntensity, CUdeviceptr scratch, size_t scratchSizeInBytes)`

See `optixDenoiserComputeIntensity()`.

## 7.24.2.12 optixDenoiserComputeMemoryResources

`OptixResult(* OptixFunctionTable::optixDenoiserComputeMemoryResources)`  
 (const `OptixDenoiser` handle, unsigned int maximumInputWidth, unsigned int maximumInputHeight, `OptixDenoiserSizes` \*returnSizes)

See `optixDenoiserComputeMemoryResources()`.

## 7.24.2.13 optixDenoiserCreate

`OptixResult(* OptixFunctionTable::optixDenoiserCreate)` (`OptixDeviceContext` context, `OptixDenoiserModelKind` modelKind, const `OptixDenoiserOptions` \*options, `OptixDenoiser` \*returnHandle)

See `optixDenoiserCreate()`.

## 7.24.2.14 optixDenoiserCreateWithUserModel

`OptixResult(* OptixFunctionTable::optixDenoiserCreateWithUserModel)`  
 (`OptixDeviceContext` context, const void \*data, size\_t dataSizeInBytes, `OptixDenoiser` \*returnHandle)

See `optixDenoiserCreateWithUserModel()`.

## 7.24.2.15 optixDenoiserDestroy

`OptixResult(* OptixFunctionTable::optixDenoiserDestroy)` (`OptixDenoiser` handle)

See `optixDenoiserDestroy()`.

## 7.24.2.16 optixDenoiserInvoke

`OptixResult(* OptixFunctionTable::optixDenoiserInvoke)` (`OptixDenoiser` denoiser, `CUstream` stream, const `OptixDenoiserParams` \*params, `CUdeviceptr` denoiserState, size\_t denoiserStateSizeInBytes, const `OptixDenoiserGuideLayer` \*guideLayer, const `OptixDenoiserLayer` \*layers, unsigned int numLayers, unsigned int inputOffsetX, unsigned int inputOffsetY, `CUdeviceptr` scratch, size\_t scratchSizeInBytes)

See `optixDenoiserInvoke()`.

## 7.24.2.17 optixDenoiserSetup

`OptixResult(* OptixFunctionTable::optixDenoiserSetup)` (`OptixDenoiser` denoiser, `CUstream` stream, unsigned int inputWidth, unsigned int inputHeight, `CUdeviceptr` state, size\_t stateSizeInBytes, `CUdeviceptr` scratch, size\_t scratchSizeInBytes)

See `optixDenoiserSetup()`.

## 7.24.2.18 optixDeviceContextCreate

`OptixResult(* OptixFunctionTable::optixDeviceContextCreate)` (`CUcontext` fromContext, const `OptixDeviceContextOptions` \*options, `OptixDeviceContext` \*context)

See `optixDeviceContextCreate()`.

#### 7.24.2.19 optixDeviceContextDestroy

`OptixResult(* OptixFunctionTable::optixDeviceContextDestroy)`  
(`OptixDeviceContext` context)

See `optixDeviceContextDestroy()`.

#### 7.24.2.20 optixDeviceContextGetCacheDatabaseSizes

`OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheDatabaseSizes)`  
(`OptixDeviceContext` context, `size_t` \*lowWaterMark, `size_t` \*highWaterMark)

See `optixDeviceContextGetCacheDatabaseSizes()`.

#### 7.24.2.21 optixDeviceContextGetCacheEnabled

`OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheEnabled)`  
(`OptixDeviceContext` context, `int` \*enabled)

See `optixDeviceContextGetCacheEnabled()`.

#### 7.24.2.22 optixDeviceContextGetCacheLocation

`OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheLocation)`  
(`OptixDeviceContext` context, `char` \*location, `size_t` locationSize)

See `optixDeviceContextGetCacheLocation()`.

#### 7.24.2.23 optixDeviceContextGetProperty

`OptixResult(* OptixFunctionTable::optixDeviceContextGetProperty)`  
(`OptixDeviceContext` context, `OptixDeviceProperty` property, `void` \*value, `size_t` sizeInBytes)

See `optixDeviceContextGetProperty()`.

#### 7.24.2.24 optixDeviceContextSetCacheDatabaseSizes

`OptixResult(* OptixFunctionTable::optixDeviceContextSetCacheDatabaseSizes)`  
(`OptixDeviceContext` context, `size_t` lowWaterMark, `size_t` highWaterMark)

See `optixDeviceContextSetCacheDatabaseSizes()`.

#### 7.24.2.25 optixDeviceContextSetCacheEnabled

`OptixResult(* OptixFunctionTable::optixDeviceContextSetCacheEnabled)`  
(`OptixDeviceContext` context, `int` enabled)

See `optixDeviceContextSetCacheEnabled()`.

#### 7.24.2.26 optixDeviceContextSetCacheLocation

`OptixResult(* OptixFunctionTable::optixDeviceContextSetCacheLocation)`  
(`OptixDeviceContext` context, `const char` \*location)

See `optixDeviceContextSetCacheLocation()`.

#### 7.24.2.27 optixDeviceContextSetLogCallback

`OptixResult(* OptixFunctionTable::optixDeviceContextSetLogCallback)`

([OptixDeviceContext](#) context, [OptixLogCallback](#) callbackFunction, void \*callbackData, unsigned int callbackLevel)

See [optixDeviceContextSetLogCallback\(\)](#).

#### 7.24.2.28 optixDisplacementMicromapArrayBuild

[OptixResult](#)(\* [OptixFunctionTable::optixDisplacementMicromapArrayBuild](#)) ([OptixDeviceContext](#) context, [CUstream](#) stream, const [OptixDisplacementMicromapArrayBuildInput](#) \*buildInput, const [OptixMicromapBuffers](#) \*buffers)

See [optixDisplacementMicromapArrayBuild\(\)](#).

#### 7.24.2.29 optixDisplacementMicromapArrayComputeMemoryUsage

[OptixResult](#)(\* [OptixFunctionTable::optixDisplacementMicromapArrayComputeMemoryUsage](#)) ([OptixDeviceContext](#) context, const [OptixDisplacementMicromapArrayBuildInput](#) \*buildInput, [OptixMicromapBufferSizes](#) \*bufferSizes)

See [optixDisplacementMicromapArrayComputeMemoryUsage\(\)](#).

#### 7.24.2.30 optixGetErrorName

const char \*(\* [OptixFunctionTable::optixGetErrorName](#)) ([OptixResult](#) result)

See [optixGetErrorName\(\)](#).

#### 7.24.2.31 optixGetErrorString

const char \*(\* [OptixFunctionTable::optixGetErrorString](#)) ([OptixResult](#) result)

See [optixGetErrorString\(\)](#).

#### 7.24.2.32 optixLaunch

[OptixResult](#)(\* [OptixFunctionTable::optixLaunch](#)) ([OptixPipeline](#) pipeline, [CUstream](#) stream, [CUdeviceptr](#) pipelineParams, size\_t pipelineParamsSize, const [OptixShaderBindingTable](#) \*sbt, unsigned int width, unsigned int height, unsigned int depth)

See [optixConvertPointerToTraversableHandle\(\)](#).

#### 7.24.2.33 optixModuleCreate

[OptixResult](#)(\* [OptixFunctionTable::optixModuleCreate](#)) ([OptixDeviceContext](#) context, const [OptixModuleCompileOptions](#) \*moduleCompileOptions, const [OptixPipelineCompileOptions](#) \*pipelineCompileOptions, const char \*input, size\_t inputSize, char \*logString, size\_t \*logStringSize, [OptixModule](#) \*module)

See [optixModuleCreate\(\)](#).

#### 7.24.2.34 optixModuleCreateWithTasks

[OptixResult](#)(\* [OptixFunctionTable::optixModuleCreateWithTasks](#)) ([OptixDeviceContext](#) context, const [OptixModuleCompileOptions](#) \*moduleCompileOptions, const [OptixPipelineCompileOptions](#) \*pipelineCompileOptions, const char \*input, size\_t inputSize, char

```
*logString, size_t *logStringSize, OptixModule *module, OptixTask
*firstTask)
```

See `optixModuleCreateWithTasks()`.

#### 7.24.2.35 optixModuleDestroy

```
OptixResult(* OptixFunctionTable::optixModuleDestroy) (OptixModule module)
```

See `optixModuleDestroy()`.

#### 7.24.2.36 optixModuleGetCompilationState

```
OptixResult(* OptixFunctionTable::optixModuleGetCompilationState)
(OptixModule module, OptixModuleCompileState *state)
```

See `optixModuleGetCompilationState()`.

#### 7.24.2.37 optixOpacityMicromapArrayBuild

```
OptixResult(* OptixFunctionTable::optixOpacityMicromapArrayBuild)
(OptixDeviceContext context, CUstream stream, const
OptixOpacityMicromapArrayBuildInput *buildInput, const OptixMicromapBuffers
*buffers)
```

See `optixOpacityMicromapArrayBuild()`.

#### 7.24.2.38 optixOpacityMicromapArrayComputeMemoryUsage

```
OptixResult(* OptixFunctionTable
::optixOpacityMicromapArrayComputeMemoryUsage) (OptixDeviceContext context,
const OptixOpacityMicromapArrayBuildInput *buildInput,
OptixMicromapBufferSizes *bufferSizes)
```

See `optixOpacityMicromapArrayComputeMemoryUsage()`.

#### 7.24.2.39 optixOpacityMicromapArrayGetRelocationInfo

```
OptixResult(* OptixFunctionTable
::optixOpacityMicromapArrayGetRelocationInfo) (OptixDeviceContext context,
CUdeviceptr opacityMicromapArray, OptixRelocationInfo *info)
```

See `optixOpacityMicromapArrayGetRelocationInfo()`.

#### 7.24.2.40 optixOpacityMicromapArrayRelocate

```
OptixResult(* OptixFunctionTable::optixOpacityMicromapArrayRelocate)
(OptixDeviceContext context, CUstream stream, const OptixRelocationInfo
*info, CUdeviceptr targetOpacityMicromapArray, size_t
targetOpacityMicromapArraySizeInBytes)
```

See `optixOpacityMicromapArrayRelocate()`.

#### 7.24.2.41 optixPipelineCreate

```
OptixResult(* OptixFunctionTable::optixPipelineCreate) (OptixDeviceContext
context, const OptixPipelineCompileOptions *pipelineCompileOptions, const
OptixPipelineLinkOptions *pipelineLinkOptions, const OptixProgramGroup
*programGroups, unsigned int numProgramGroups, char *logString, size_t
```

`*logStringSize, OptixPipeline *pipeline)`

See `optixPipelineCreate()`.

#### 7.24.2.42 `optixPipelineDestroy`

`OptixResult(* OptixFunctionTable::optixPipelineDestroy) (OptixPipeline pipeline)`

See `optixPipelineDestroy()`.

#### 7.24.2.43 `optixPipelineSetStackSize`

`OptixResult(* OptixFunctionTable::optixPipelineSetStackSize) (OptixPipeline pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)`

See `optixPipelineSetStackSize()`.

#### 7.24.2.44 `optixProgramGroupCreate`

`OptixResult(* OptixFunctionTable::optixProgramGroupCreate) (OptixDeviceContext context, const OptixProgramGroupDesc *programDescriptions, unsigned int numProgramGroups, const OptixProgramGroupOptions *options, char *logString, size_t *logStringSize, OptixProgramGroup *programGroups)`

See `optixProgramGroupCreate()`.

#### 7.24.2.45 `optixProgramGroupDestroy`

`OptixResult(* OptixFunctionTable::optixProgramGroupDestroy) (OptixProgramGroup programGroup)`

See `optixProgramGroupDestroy()`.

#### 7.24.2.46 `optixProgramGroupGetStackSize`

`OptixResult(* OptixFunctionTable::optixProgramGroupGetStackSize) (OptixProgramGroup programGroup, OptixStackSizes *stackSizes, OptixPipeline pipeline)`

See `optixProgramGroupGetStackSize()`.

#### 7.24.2.47 `optixSbtRecordPackHeader`

`OptixResult(* OptixFunctionTable::optixSbtRecordPackHeader) (OptixProgramGroup programGroup, void *sbtRecordHeaderHostPointer)`

See `optixConvertPointerToTraversableHandle()`.

#### 7.24.2.48 `optixTaskExecute`

`OptixResult(* OptixFunctionTable::optixTaskExecute) (OptixTask task, OptixTask *additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int *numAdditionalTasksCreated)`

See `optixTaskExecute()`.



## 7.25 OptixImage2D Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- [CUdeviceptr](#) `data`
- unsigned int `width`
- unsigned int `height`
- unsigned int `rowStrideInBytes`
- unsigned int `pixelStrideInBytes`
- [OptixPixelFormat](#) `format`

### 7.25.1 Detailed Description

Image descriptor used by the denoiser.

See also [optixDenoiserInvoke\(\)](#), [optixDenoiserComputeIntensity\(\)](#)

### 7.25.2 Member Data Documentation

#### 7.25.2.1 `data`

[CUdeviceptr](#) `OptixImage2D::data`

Pointer to the actual pixel data.

#### 7.25.2.2 `format`

[OptixPixelFormat](#) `OptixImage2D::format`

Pixel format.

#### 7.25.2.3 `height`

unsigned int `OptixImage2D::height`

Height of the image (in pixels)

#### 7.25.2.4 `pixelStrideInBytes`

unsigned int `OptixImage2D::pixelStrideInBytes`

Stride between subsequent pixels of the image (in bytes). If set to 0, dense packing (no gaps) is assumed. For pixel format `OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER` it must be set to [OptixDenoiserSizes::internalGuideLayerPixelSizeInBytes](#).

#### 7.25.2.5 `rowStrideInBytes`

unsigned int `OptixImage2D::rowStrideInBytes`

Stride between subsequent rows of the image (in bytes).

#### 7.25.2.6 `width`

unsigned int `OptixImage2D::width`

Width of the image (in pixels)



## 7.26 OptixInstance Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- float [transform](#) [12]
- unsigned int [instanceId](#)
- unsigned int [sbtOffset](#)
- unsigned int [visibilityMask](#)
- unsigned int [flags](#)
- [OptixTraversableHandle](#) [traversableHandle](#)
- unsigned int [pad](#) [2]

### 7.26.1 Detailed Description

Instances.

See also [OptixBuildInputInstanceArray::instances](#)

### 7.26.2 Member Data Documentation

#### 7.26.2.1 flags

```
unsigned int OptixInstance::flags
```

Any combination of [OptixInstanceFlags](#) is allowed.

#### 7.26.2.2 instanceId

```
unsigned int OptixInstance::instanceId
```

Application supplied ID. The maximal ID can be queried using [OPTIX\\_DEVICE\\_PROPERTY\\_LIMIT\\_MAX\\_INSTANCE\\_ID](#).

#### 7.26.2.3 pad

```
unsigned int OptixInstance::pad[2]
```

round up to 80-byte, to ensure 16-byte alignment

#### 7.26.2.4 sbtOffset

```
unsigned int OptixInstance::sbtOffset
```

SBT record offset. In a traversable graph with multiple levels of instance acceleration structure (IAS) objects, offsets are summed together. The maximal SBT offset can be queried using [OPTIX\\_DEVICE\\_PROPERTY\\_LIMIT\\_MAX\\_SBT\\_OFFSET](#).

#### 7.26.2.5 transform

```
float OptixInstance::transform[12]
```

affine object-to-world transformation as 3x4 matrix in row-major layout

#### 7.26.2.6 traversableHandle

```
OptixTraversableHandle OptixInstance::traversableHandle
```

Set with an [OptixTraversableHandle](#).

### 7.26.2.7 visibilityMask

unsigned int OptixInstance::visibilityMask

Visibility mask. If rayMask & instanceMask == 0 the instance is culled. The number of available bits can be queried using OPTIX\_DEVICE\_PROPERTY\_LIMIT\_NUM\_BITS\_INSTANCE\_VISIBILITY\_MASK.

## 7.27 OptixMatrixMotionTransform Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- [OptixTraversableHandle](#) child
- [OptixMotionOptions](#) motionOptions
- unsigned int pad [3]
- float transform [2][12]

### 7.27.1 Detailed Description

Represents a matrix motion transformation.

The device address of instances of this type must be a multiple of OPTIX\_TRANSFORM\_BYTE\_ALIGNMENT.

This struct, as defined here, handles only N=2 motion keys due to the fixed array length of its transform member. The following example shows how to create instances for an arbitrary number N of motion keys:

```
float matrixData[N][12];
... // setup matrixData
size_t transformSizeInBytes = sizeof(OptixMatrixMotionTransform) + (N-2) * 12 * sizeof(float);
OptixMatrixMotionTransform* matrixMoptionTransform = (OptixMatrixMotionTransform*)
malloc(transformSizeInBytes);
memset(matrixMoptionTransform, 0, transformSizeInBytes);
... // setup other members of matrixMoptionTransform
matrixMoptionTransform->motionOptions.numKeys
memcpy(matrixMoptionTransform->transform, matrixData, N * 12 * sizeof(float));
... // copy matrixMoptionTransform to device memory
free(matrixMoptionTransform)
```

See also [optixConvertPointerToTraversableHandle\(\)](#)

### 7.27.2 Member Data Documentation

#### 7.27.2.1 child

[OptixTraversableHandle](#) OptixMatrixMotionTransform::child

The traversable that is transformed by this transformation.

#### 7.27.2.2 motionOptions

[OptixMotionOptions](#) OptixMatrixMotionTransform::motionOptions

The motion options for this transformation. Must have at least two motion keys.

#### 7.27.2.3 pad

unsigned int OptixMatrixMotionTransform::pad[3]

Padding to make the transformation 16 byte aligned.

#### 7.27.2.4 transform

`float OptixMatrixMotionTransform::transform[2][12]`

Affine object-to-world transformation as 3x4 matrix in row-major layout.

### 7.28 OptixMicromapBuffers Struct Reference

`#include <optix_types.h>`

#### Public Attributes

- `CUdeviceptr output`
- `size_t outputSizeInBytes`
- `CUdeviceptr temp`
- `size_t tempSizeInBytes`

#### 7.28.1 Detailed Description

Buffer inputs for opacity/displacement micromap array builds.

#### 7.28.2 Member Data Documentation

##### 7.28.2.1 output

`CUdeviceptr OptixMicromapBuffers::output`

Output buffer.

##### 7.28.2.2 outputSizeInBytes

`size_t OptixMicromapBuffers::outputSizeInBytes`

Output buffer size.

##### 7.28.2.3 temp

`CUdeviceptr OptixMicromapBuffers::temp`

Temp buffer.

##### 7.28.2.4 tempSizeInBytes

`size_t OptixMicromapBuffers::tempSizeInBytes`

Temp buffer size.

### 7.29 OptixMicromapBufferSizes Struct Reference

`#include <optix_types.h>`

#### Public Attributes

- `size_t outputSizeInBytes`
- `size_t tempSizeInBytes`

#### 7.29.1 Detailed Description

Conservative memory requirements for building a opacity/displacement micromap array.

## 7.29.2 Member Data Documentation

### 7.29.2.1 outputSizeInBytes

`size_t OptixMicromapBufferSizes::outputSizeInBytes`

### 7.29.2.2 tempSizeInBytes

`size_t OptixMicromapBufferSizes::tempSizeInBytes`

## 7.30 OptixModuleCompileBoundValueEntry Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- `size_t pipelineParamOffsetInBytes`
- `size_t sizeInBytes`
- `const void * boundValuePtr`
- `const char * annotation`

### 7.30.1 Detailed Description

Struct for specifying specializations for pipelineParams as specified in [OptixPipelineCompileOptions::pipelineLaunchParamsVariableName](#).

The bound values are supposed to represent a constant value in the pipelineParams. OptiX will attempt to locate all loads from the pipelineParams and correlate them to the appropriate bound value, but there are cases where OptiX cannot safely or reliably do this. For example if the pointer to the pipelineParams is passed as an argument to a non-inline function or the offset of the load to the pipelineParams cannot be statically determined (e.g. accessed in a loop). No module should rely on the value being specialized in order to work correctly. The values in the pipelineParams specified on `optixLaunch` should match the bound value. If validation mode is enabled on the context, OptiX will verify that the bound values specified matches the values in pipelineParams specified to `optixLaunch`.

These values are compiled in to the module as constants. Once the constants are inserted into the code, an optimization pass will be run that will attempt to propagate the constants and remove unreachable code.

If caching is enabled, changes in these values will result in newly compiled modules.

The `pipelineParamOffset` and `sizeInBytes` must be within the bounds of the pipelineParams variable. `OPTIX_ERROR_INVALID_VALUE` will be returned from `optixModuleCreate` otherwise.

If more than one bound value overlaps or the size of a bound value is equal to 0, an `OPTIX_ERROR_INVALID_VALUE` will be returned from `optixModuleCreate`.

The same set of bound values do not need to be used for all modules in a pipeline, but overlapping values between modules must have the same value. `OPTIX_ERROR_INVALID_VALUE` will be returned from `optixPipelineCreate` otherwise.

See also [OptixModuleCompileOptions](#)

## 7.30.2 Member Data Documentation

### 7.30.2.1 annotation

`const char* OptixModuleCompileBoundValueEntry::annotation`

### 7.30.2.2 boundValuePtr

`const void* OptixModuleCompileBoundValueEntry::boundValuePtr`

### 7.30.2.3 pipelineParamOffsetInBytes

`size_t OptixModuleCompileBoundValueEntry::pipelineParamOffsetInBytes`

### 7.30.2.4 sizeInBytes

`size_t OptixModuleCompileBoundValueEntry::sizeInBytes`

## 7.31 OptixModuleCompileOptions Struct Reference

`#include <optix_types.h>`

### Public Attributes

- `int maxRegisterCount`
- `OptixCompileOptimizationLevel optLevel`
- `OptixCompileDebugLevel debugLevel`
- `const OptixModuleCompileBoundValueEntry * boundValues`
- `unsigned int numBoundValues`
- `unsigned int numPayloadTypes`
- `OptixPayloadType * payloadTypes`

### 7.31.1 Detailed Description

Compilation options for module.

See also `optixModuleCreate()`

### 7.31.2 Member Data Documentation

#### 7.31.2.1 boundValues

`const OptixModuleCompileBoundValueEntry* OptixModuleCompileOptions::boundValues`

Ignored if numBoundValues is set to 0.

#### 7.31.2.2 debugLevel

`OptixCompileDebugLevel OptixModuleCompileOptions::debugLevel`

Generate debug information.

#### 7.31.2.3 maxRegisterCount

`int OptixModuleCompileOptions::maxRegisterCount`

Maximum number of registers allowed when compiling to SASS. Set to 0 for no explicit limit. May vary within a pipeline.

#### 7.31.2.4 numBoundValues

`unsigned int OptixModuleCompileOptions::numBoundValues`

set to 0 if unused

### 7.31.2.5 numPayloadTypes

unsigned int OptixModuleCompileOptions::numPayloadTypes

The number of different payload types available for compilation. Must be zero if [OptixPipelineCompileOptions::numPayloadValues](#) is not zero.

### 7.31.2.6 optLevel

[OptixCompileOptimizationLevel](#) OptixModuleCompileOptions::optLevel

Optimization level. May vary within a pipeline.

### 7.31.2.7 payloadTypes

[OptixPayloadType\\*](#) OptixModuleCompileOptions::payloadTypes

Points to host array of payload type definitions, size must match numPayloadTypes.

## 7.32 OptixMotionOptions Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- unsigned short [numKeys](#)
- unsigned short [flags](#)
- float [timeBegin](#)
- float [timeEnd](#)

### 7.32.1 Detailed Description

Motion options.

See also [OptixAccelBuildOptions::motionOptions](#), [OptixMatrixMotionTransform::motionOptions](#), [OptixSRTMotionTransform::motionOptions](#)

### 7.32.2 Member Data Documentation

#### 7.32.2.1 flags

unsigned short OptixMotionOptions::flags

Combinations of [OptixMotionFlags](#).

#### 7.32.2.2 numKeys

unsigned short OptixMotionOptions::numKeys

If numKeys > 1, motion is enabled. timeBegin, timeEnd and flags are all ignored when motion is disabled.

#### 7.32.2.3 timeBegin

float OptixMotionOptions::timeBegin

Point in time where motion starts. Must be lesser than timeEnd.

#### 7.32.2.4 timeEnd

`float OptixMotionOptions::timeEnd`

Point in time where motion ends. Must be greater than timeBegin.

### 7.33 OptixOpacityMicromapArrayBuildInput Struct Reference

`#include <optix_types.h>`

#### Public Attributes

- unsigned int `flags`
- `CUdeviceptr` `inputBuffer`
- `CUdeviceptr` `perMicromapDescBuffer`
- unsigned int `perMicromapDescStrideInBytes`
- unsigned int `numMicromapHistogramEntries`
- const `OptixOpacityMicromapHistogramEntry` \* `micromapHistogramEntries`

#### 7.33.1 Detailed Description

Inputs to opacity micromap array construction.

#### 7.33.2 Member Data Documentation

##### 7.33.2.1 flags

`unsigned int OptixOpacityMicromapArrayBuildInput::flags`

Applies to all opacity micromaps in array.

##### 7.33.2.2 inputBuffer

`CUdeviceptr OptixOpacityMicromapArrayBuildInput::inputBuffer`

128B aligned base pointer for raw opacity micromap input data.

##### 7.33.2.3 micromapHistogramEntries

`const OptixOpacityMicromapHistogramEntry*`  
`OptixOpacityMicromapArrayBuildInput::micromapHistogramEntries`

Histogram over opacity micromaps of input format and subdivision combinations. Counts of entries with equal format and subdivision combination (duplicates) are added together.

##### 7.33.2.4 numMicromapHistogramEntries

`unsigned int OptixOpacityMicromapArrayBuildInput`  
`::numMicromapHistogramEntries`

Number of `OptixOpacityMicromapHistogramEntry`.

##### 7.33.2.5 perMicromapDescBuffer

`CUdeviceptr OptixOpacityMicromapArrayBuildInput::perMicromapDescBuffer`

One `OptixOpacityMicromapDesc` entry per opacity micromap. This device pointer must be a multiple of `OPTIX_OPACITY_MICROMAP_DESC_BYTE_ALIGNMENT`.

### 7.33.2.6 perMicromapDescStrideInBytes

```
unsigned int OptixOpacityMicromapArrayBuildInput
::perMicromapDescStrideInBytes
```

Stride between OptixOpacityMicromapDescs in perOmDescBuffer. If set to zero, the opacity micromap descriptors are assumed to be tightly packed and the stride is assumed to be `sizeof(OptixOpacityMicromapDesc)`. This stride must be a multiple of `OPTIX_OPACITY_MICROMAP_DESC_BYTE_ALIGNMENT`.

## 7.34 OptixOpacityMicromapDesc Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- unsigned int [byteOffset](#)
- unsigned short [subdivisionLevel](#)
- unsigned short [format](#)

### 7.34.1 Detailed Description

Opacity micromap descriptor.

### 7.34.2 Member Data Documentation

#### 7.34.2.1 byteOffset

```
unsigned int OptixOpacityMicromapDesc::byteOffset
```

Byte offset to opacity micromap in data input buffer of opacity micromap array build.

#### 7.34.2.2 format

```
unsigned short OptixOpacityMicromapDesc::format
```

OptixOpacityMicromapFormat.

#### 7.34.2.3 subdivisionLevel

```
unsigned short OptixOpacityMicromapDesc::subdivisionLevel
```

Number of micro-triangles is  $4^{\text{level}}$ . Valid levels are [0, 12].

## 7.35 OptixOpacityMicromapHistogramEntry Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- unsigned int [count](#)
- unsigned int [subdivisionLevel](#)
- [OptixOpacityMicromapFormat](#) [format](#)

### 7.35.1 Detailed Description

Opacity micromap histogram entry. Specifies how many opacity micromaps of a specific type are input to the opacity micromap array build. Note that while this is similar to



[OptixOpacityMicromapUsageCount](#), the histogram entry specifies how many opacity micromaps of a specific type are combined into a opacity micromap array.

## 7.35.2 Member Data Documentation

### 7.35.2.1 count

`unsigned int OptixOpacityMicromapHistogramEntry::count`

Number of opacity micromaps with the format and subdivision level that are input to the opacity micromap array build.

### 7.35.2.2 format

`OptixOpacityMicromapFormat OptixOpacityMicromapHistogramEntry::format`

Opacity micromap format.

### 7.35.2.3 subdivisionLevel

`unsigned int OptixOpacityMicromapHistogramEntry::subdivisionLevel`

Number of micro-triangles is  $4^{\text{level}}$ . Valid levels are [0, 12].

## 7.36 OptixOpacityMicromapUsageCount Struct Reference

`#include <optix_types.h>`

### Public Attributes

- unsigned int `count`
- unsigned int `subdivisionLevel`
- `OptixOpacityMicromapFormat` `format`

### 7.36.1 Detailed Description

Opacity micromap usage count for acceleration structure builds. Specifies how many opacity micromaps of a specific type are referenced by triangles when building the AS. Note that while this is similar to [OptixOpacityMicromapHistogramEntry](#), the usage count specifies how many opacity micromaps of a specific type are referenced by triangles in the AS.

## 7.36.2 Member Data Documentation

### 7.36.2.1 count

`unsigned int OptixOpacityMicromapUsageCount::count`

Number of opacity micromaps with this format and subdivision level referenced by triangles in the corresponding triangle build input at AS build time.

### 7.36.2.2 format

`OptixOpacityMicromapFormat OptixOpacityMicromapUsageCount::format`

opacity micromap format.

### 7.36.2.3 subdivisionLevel

`unsigned int OptixOpacityMicromapUsageCount::subdivisionLevel`

Number of micro-triangles is  $4^{\text{level}}$ . Valid levels are [0, 12].

## 7.37 OptixPayloadType Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- unsigned int [numPayloadValues](#)
- const unsigned int \* [payloadSemantics](#)

#### 7.37.1 Detailed Description

Specifies a single payload type.

#### 7.37.2 Member Data Documentation

##### 7.37.2.1 numPayloadValues

```
unsigned int OptixPayloadType::numPayloadValues
```

The number of 32b words the payload of this type holds.

##### 7.37.2.2 payloadSemantics

```
const unsigned int* OptixPayloadType::payloadSemantics
```

Points to host array of payload word semantics, size must match numPayloadValues.

## 7.38 OptixPipelineCompileOptions Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- int [usesMotionBlur](#)
- unsigned int [traversableGraphFlags](#)
- int [numPayloadValues](#)
- int [numAttributeValues](#)
- unsigned int [exceptionFlags](#)
- const char \* [pipelineLaunchParamsVariableName](#)
- unsigned int [usesPrimitiveTypeFlags](#)
- int [allowOpacityMicromaps](#)

#### 7.38.1 Detailed Description

Compilation options for all modules of a pipeline.

Similar to [OptixModuleCompileOptions](#), but these options here need to be equal for all modules of a pipeline.

See also [optixModuleCreate\(\)](#), [optixPipelineCreate\(\)](#)

#### 7.38.2 Member Data Documentation

##### 7.38.2.1 allowOpacityMicromaps

```
int OptixPipelineCompileOptions::allowOpacityMicromaps
```

Boolean value indicating whether opacity micromaps could be used.

### 7.38.2.2 exceptionFlags

`unsigned int OptixPipelineCompileOptions::exceptionFlags`

A bitmask of `OptixExceptionFlags` indicating which exceptions are enabled.

### 7.38.2.3 numAttributeValues

`int OptixPipelineCompileOptions::numAttributeValues`

How much storage, in 32b words, to make available for the attributes. The minimum number is 2.

Values below that will automatically be changed to 2. [2..8].

### 7.38.2.4 numPayloadValues

`int OptixPipelineCompileOptions::numPayloadValues`

How much storage, in 32b words, to make available for the payload, [0..32] Must be zero if `numPayloadTypes` is not zero.

### 7.38.2.5 pipelineLaunchParamsVariableName

`const char* OptixPipelineCompileOptions::pipelineLaunchParamsVariableName`

The name of the pipeline parameter variable. If 0, no pipeline parameter will be available. This will be ignored if the launch param variable was optimized out or was not found in the modules linked to the pipeline.

### 7.38.2.6 traversableGraphFlags

`unsigned int OptixPipelineCompileOptions::traversableGraphFlags`

Traversable graph bitfield. See `OptixTraversableGraphFlags`.

### 7.38.2.7 usesMotionBlur

`int OptixPipelineCompileOptions::usesMotionBlur`

Boolean value indicating whether motion blur could be used.

### 7.38.2.8 usesPrimitiveTypeFlags

`unsigned int OptixPipelineCompileOptions::usesPrimitiveTypeFlags`

Bit field enabling primitive types. See `OptixPrimitiveTypeFlags`. Setting to zero corresponds to enabling `OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM` and `OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE`.

## 7.39 OptixPipelineLinkOptions Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- unsigned int [maxTraceDepth](#)

### 7.39.1 Detailed Description

Link options for a pipeline.

See also [optixPipelineCreate\(\)](#)

### 7.39.2 Member Data Documentation

#### 7.39.2.1 maxTraceDepth

`unsigned int OptixPipelineLinkOptions::maxTraceDepth`

Maximum trace recursion depth. 0 means a ray generation program can be launched, but can't trace any rays. The maximum allowed value is 31.

## 7.40 OptixProgramGroupCallables Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- [OptixModule moduleDC](#)
- `const char * entryFunctionNameDC`
- [OptixModule moduleCC](#)
- `const char * entryFunctionNameCC`

### 7.40.1 Detailed Description

Program group representing callables.

Module and entry function name need to be valid for at least one of the two callables.

See also [#OptixProgramGroupDesc::callables](#)

### 7.40.2 Member Data Documentation

#### 7.40.2.1 entryFunctionNameCC

`const char* OptixProgramGroupCallables::entryFunctionNameCC`

Entry function name of the continuation callable (CC) program.

#### 7.40.2.2 entryFunctionNameDC

`const char* OptixProgramGroupCallables::entryFunctionNameDC`

Entry function name of the direct callable (DC) program.

#### 7.40.2.3 moduleCC

[OptixModule](#) `OptixProgramGroupCallables::moduleCC`

Module holding the continuation callable (CC) program.

#### 7.40.2.4 moduleDC

[OptixModule](#) `OptixProgramGroupCallables::moduleDC`

Module holding the direct callable (DC) program.

## 7.41 OptixProgramGroupDesc Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- [OptixProgramGroupKind](#) kind
  - unsigned int flags
  - union {
    - [OptixProgramGroupSingleModule](#) raygen
    - [OptixProgramGroupSingleModule](#) miss
    - [OptixProgramGroupSingleModule](#) exception
    - [OptixProgramGroupCallables](#) callables
    - [OptixProgramGroupHitgroup](#) hitgroup
- ```
};
```

### 7.41.1 Detailed Description

Descriptor for program groups.

### 7.41.2 Member Data Documentation

#### 7.41.2.1

```
union { ... } OptixProgramGroupDesc::@5
```

#### 7.41.2.2 callables

[OptixProgramGroupCallables](#) [OptixProgramGroupDesc::callables](#)

See also [OPTIX\\_PROGRAM\\_GROUP\\_KIND\\_CALLABLES](#)

#### 7.41.2.3 exception

[OptixProgramGroupSingleModule](#) [OptixProgramGroupDesc::exception](#)

See also [OPTIX\\_PROGRAM\\_GROUP\\_KIND\\_EXCEPTION](#)

#### 7.41.2.4 flags

unsigned int [OptixProgramGroupDesc::flags](#)

See [OptixProgramGroupFlags](#).

#### 7.41.2.5 hitgroup

[OptixProgramGroupHitgroup](#) [OptixProgramGroupDesc::hitgroup](#)

See also [OPTIX\\_PROGRAM\\_GROUP\\_KIND\\_HITGROUP](#)

#### 7.41.2.6 kind

[OptixProgramGroupKind](#) [OptixProgramGroupDesc::kind](#)

The kind of program group.

### 7.41.2.7 miss

[OptixProgramGroupSingleModule](#) [OptixProgramGroupDesc::miss](#)

See also [OPTIX\\_PROGRAM\\_GROUP\\_KIND\\_MISS](#)

### 7.41.2.8 raygen

[OptixProgramGroupSingleModule](#) [OptixProgramGroupDesc::raygen](#)

See also [OPTIX\\_PROGRAM\\_GROUP\\_KIND\\_RAYGEN](#)

## 7.42 OptixProgramGroupHitgroup Struct Reference

`#include <optix_types.h>`

### Public Attributes

- [OptixModule](#) [moduleCH](#)
- `const char *` [entryFunctionNameCH](#)
- [OptixModule](#) [moduleAH](#)
- `const char *` [entryFunctionNameAH](#)
- [OptixModule](#) [moduleIS](#)
- `const char *` [entryFunctionNameIS](#)

### 7.42.1 Detailed Description

Program group representing the hitgroup.

For each of the three program types, module and entry function name might both be `nullptr`.

See also [OptixProgramGroupDesc::hitgroup](#)

### 7.42.2 Member Data Documentation

#### 7.42.2.1 entryFunctionNameAH

`const char*` [OptixProgramGroupHitgroup::entryFunctionNameAH](#)

Entry function name of the any hit (AH) program.

#### 7.42.2.2 entryFunctionNameCH

`const char*` [OptixProgramGroupHitgroup::entryFunctionNameCH](#)

Entry function name of the closest hit (CH) program.

#### 7.42.2.3 entryFunctionNameIS

`const char*` [OptixProgramGroupHitgroup::entryFunctionNameIS](#)

Entry function name of the intersection (IS) program.

#### 7.42.2.4 moduleAH

[OptixModule](#) [OptixProgramGroupHitgroup::moduleAH](#)

Module holding the any hit (AH) program.

### 7.42.2.5 moduleCH

**OptixModule** OptixProgramGroupHitgroup::moduleCH

Module holding the closest hit (CH) program.

### 7.42.2.6 moduleIS

**OptixModule** OptixProgramGroupHitgroup::moduleIS

Module holding the intersection (Is) program.

## 7.43 OptixProgramGroupOptions Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- **OptixPayloadType** \* payloadType

### 7.43.1 Detailed Description

Program group options.

See also [optixProgramGroupCreate\(\)](#)

### 7.43.2 Member Data Documentation

#### 7.43.2.1 payloadType

**OptixPayloadType**\* OptixProgramGroupOptions::payloadType

Specifies the payload type of this program group. All programs in the group must support the payload type (Program support for a type is specified by calling.

See also [optixSetPayloadTypes](#) or otherwise all types specified in

[OptixModuleCompileOptions](#) are supported). If a program is not available for the requested payload type, [optixProgramGroupCreate](#) returns [OPTIX\\_ERROR\\_PAYLOAD\\_TYPE\\_MISMATCH](#). If the [payloadType](#) is left zero, a unique type is deduced. The payload type can be uniquely deduced if there is exactly one payload type for which all programs in the group are available. If the payload type could not be deduced uniquely [optixProgramGroupCreate](#) returns [OPTIX\\_ERROR\\_PAYLOAD\\_TYPE\\_RESOLUTION\\_FAILED](#).

## 7.44 OptixProgramGroupSingleModule Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- **OptixModule** module
- const char \* entryFunctionName

### 7.44.1 Detailed Description

Program group representing a single module.

Used for raygen, miss, and exception programs. In case of raygen and exception programs, module and entry function name need to be valid. For miss programs, module and entry function name might both be `nullptr`.

See also [OptixProgramGroupDesc::raygen](#), [OptixProgramGroupDesc::miss](#), [OptixProgramGroupDesc::exception](#)

## 7.44.2 Member Data Documentation

### 7.44.2.1 entryFunctionName

`const char* OptixProgramGroupSingleModule::entryFunctionName`

Entry function name of the single program.

### 7.44.2.2 module

[OptixModule](#) `OptixProgramGroupSingleModule::module`

Module holding single program.

## 7.45 OptixRelocateInput Struct Reference

`#include <optix_types.h>`

### Public Attributes

- [OptixBuildInputType](#) type
- union {
  - [OptixRelocateInputInstanceArray](#) instanceArray
  - [OptixRelocateInputTriangleArray](#) triangleArray
- };

### 7.45.1 Detailed Description

Relocation inputs.

See also [optixAccelRelocate\(\)](#)

## 7.45.2 Member Data Documentation

### 7.45.2.1

`union { ... } OptixRelocateInput::@3`

### 7.45.2.2 instanceArray

[OptixRelocateInputInstanceArray](#) `OptixRelocateInput::instanceArray`

Instance and instance pointer inputs.

### 7.45.2.3 triangleArray

[OptixRelocateInputTriangleArray](#) `OptixRelocateInput::triangleArray`

Triangle inputs.

### 7.45.2.4 type

[OptixBuildInputType](#) `OptixRelocateInput::type`

The type of the build input to relocate.



## 7.46 OptixRelocateInputInstanceArray Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- unsigned int [numInstances](#)
- [CUdeviceptr](#) [traversableHandles](#)

#### 7.46.1 Detailed Description

Instance and instance pointer inputs.

See also [OptixRelocateInput::instanceArray](#)

#### 7.46.2 Member Data Documentation

##### 7.46.2.1 numInstances

```
unsigned int OptixRelocateInputInstanceArray::numInstances
```

Number of elements in [OptixRelocateInputInstanceArray::traversableHandles](#). Must match [OptixBuildInputInstanceArray::numInstances](#) of the source build input.

##### 7.46.2.2 traversableHandles

```
CUdeviceptr OptixRelocateInputInstanceArray::traversableHandles
```

These are the traversable handles of the instances (See [OptixInstance::traversableHandle](#)) These can be used when also relocating the instances. No updates to the bounds are performed. Use [optixAccelBuild](#) to update the bounds. 'traversableHandles' may be zero when the traversables are not relocated (i.e. relocation of an IAS on the source device).

## 7.47 OptixRelocateInputOpacityMicromap Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- [CUdeviceptr](#) [opacityMicromapArray](#)

#### 7.47.1 Member Data Documentation

##### 7.47.1.1 opacityMicromapArray

```
CUdeviceptr OptixRelocateInputOpacityMicromap::opacityMicromapArray
```

Device pointer to a relocated opacity micromap array used by the source build input array. May be zero when no micromaps were used in the source accel, or the referenced opacity micromaps don't require relocation (for example relocation of a GAS on the source device).

## 7.48 OptixRelocateInputTriangleArray Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- unsigned int [numSbtRecords](#)
- [OptixRelocateInputOpacityMicromap](#) [opacityMicromap](#)

### 7.48.1 Detailed Description

Triangle inputs.

See also [OptixRelocateInput::triangleArray](#)

### 7.48.2 Member Data Documentation

#### 7.48.2.1 numSbtRecords

`unsigned int OptixRelocateInputTriangleArray::numSbtRecords`

Number of sbt records available to the sbt index offset override. Must match [OptixBuildInputTriangleArray::numSbtRecords](#) of the source build input.

#### 7.48.2.2 opacityMicromap

`OptixRelocateInputOpacityMicromap OptixRelocateInputTriangleArray::opacityMicromap`

Opacity micromap inputs.

## 7.49 OptixRelocationInfo Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- `unsigned long long info [4]`

### 7.49.1 Detailed Description

Used to store information related to relocation of optix data structures.

See also [optixOpacityMicromapArrayGetRelocationInfo\(\)](#), [optixOpacityMicromapArrayRelocate\(\)](#), [optixAccelGetRelocationInfo\(\)](#), [optixAccelRelocate\(\)](#), [optixCheckRelocationCompatibility\(\)](#)

### 7.49.2 Member Data Documentation

#### 7.49.2.1 info

`unsigned long long OptixRelocationInfo::info[4]`

Opaque data, used internally, should not be modified.

## 7.50 OptixShaderBindingTable Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- `CUdeviceptr raygenRecord`
- `CUdeviceptr exceptionRecord`
- `CUdeviceptr missRecordBase`
- `unsigned int missRecordStrideInBytes`
- `unsigned int missRecordCount`
- `CUdeviceptr hitgroupRecordBase`

- unsigned int `hitgroupRecordStrideInBytes`
- unsigned int `hitgroupRecordCount`
- `CUdeviceptr` `callablesRecordBase`
- unsigned int `callablesRecordStrideInBytes`
- unsigned int `callablesRecordCount`

### 7.50.1 Detailed Description

Describes the shader binding table (SBT)

See also `optixLaunch()`

### 7.50.2 Member Data Documentation

#### 7.50.2.1 `callablesRecordBase`

`CUdeviceptr` `OptixShaderBindingTable::callablesRecordBase`

Arrays of SBT records for callable programs. If the base address is not null, the stride and count must not be zero. If the base address is null, then the count needs to zero. The base address and the stride must be a multiple of `OPTIX_SBT_RECORD_ALIGNMENT`.

#### 7.50.2.2 `callablesRecordCount`

unsigned int `OptixShaderBindingTable::callablesRecordCount`

Arrays of SBT records for callable programs. If the base address is not null, the stride and count must not be zero. If the base address is null, then the count needs to zero. The base address and the stride must be a multiple of `OPTIX_SBT_RECORD_ALIGNMENT`.

#### 7.50.2.3 `callablesRecordStrideInBytes`

unsigned int `OptixShaderBindingTable::callablesRecordStrideInBytes`

Arrays of SBT records for callable programs. If the base address is not null, the stride and count must not be zero. If the base address is null, then the count needs to zero. The base address and the stride must be a multiple of `OPTIX_SBT_RECORD_ALIGNMENT`.

#### 7.50.2.4 `exceptionRecord`

`CUdeviceptr` `OptixShaderBindingTable::exceptionRecord`

Device address of the SBT record of the exception program. The address must be a multiple of `OPTIX_SBT_RECORD_ALIGNMENT`.

#### 7.50.2.5 `hitgroupRecordBase`

`CUdeviceptr` `OptixShaderBindingTable::hitgroupRecordBase`

Arrays of SBT records for hit groups. The base address and the stride must be a multiple of `OPTIX_SBT_RECORD_ALIGNMENT`.

#### 7.50.2.6 `hitgroupRecordCount`

unsigned int `OptixShaderBindingTable::hitgroupRecordCount`

Arrays of SBT records for hit groups. The base address and the stride must be a multiple of `OPTIX_SBT_RECORD_ALIGNMENT`.

### 7.50.2.7 hitgroupRecordStrideInBytes

`unsigned int OptixShaderBindingTable::hitgroupRecordStrideInBytes`

Arrays of SBT records for hit groups. The base address and the stride must be a multiple of `OPTIX_SBT_RECORD_ALIGNMENT`.

### 7.50.2.8 missRecordBase

`CUdeviceptr OptixShaderBindingTable::missRecordBase`

Arrays of SBT records for miss programs. The base address and the stride must be a multiple of `OPTIX_SBT_RECORD_ALIGNMENT`.

### 7.50.2.9 missRecordCount

`unsigned int OptixShaderBindingTable::missRecordCount`

Arrays of SBT records for miss programs. The base address and the stride must be a multiple of `OPTIX_SBT_RECORD_ALIGNMENT`.

### 7.50.2.10 missRecordStrideInBytes

`unsigned int OptixShaderBindingTable::missRecordStrideInBytes`

Arrays of SBT records for miss programs. The base address and the stride must be a multiple of `OPTIX_SBT_RECORD_ALIGNMENT`.

### 7.50.2.11 raygenRecord

`CUdeviceptr OptixShaderBindingTable::raygenRecord`

Device address of the SBT record of the ray gen program to start launch at. The address must be a multiple of `OPTIX_SBT_RECORD_ALIGNMENT`.

## 7.51 OptixSRTData Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

Parameters describing the SRT transformation

- `float sx`
- `float a`
- `float b`
- `float pvx`
- `float sy`
- `float c`
- `float pvy`
- `float sz`
- `float pvz`
- `float qx`
- `float qy`
- `float qz`
- `float qw`
- `float tx`
- `float ty`
- `float tz`

### 7.51.1 Detailed Description

Represents an SRT transformation.

An SRT transformation can represent a smooth rotation with fewer motion keys than a matrix transformation. Each motion key is constructed from elements taken from a matrix  $S$ , a quaternion  $R$ , and a translation  $T$ .

The scaling matrix  $S = \begin{bmatrix} sx & a & b & pvx \\ 0 & sy & c & pvy \\ 0 & 0 & sz & pvz \end{bmatrix}$  defines an affine transformation that can include scale,

shear, and a translation. The translation allows to define the pivot point for the subsequent rotation.

The quaternion  $R = [qx, qy, qz, qw]$  describes a rotation with angular component  $qw = \cos(\theta/2)$  and other components  $[qx, qy, qz] = \sin(\theta/2) * [ax, ay, az]$  where the axis  $[ax, ay, az]$  is normalized.

The translation matrix  $T = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \end{bmatrix}$  defines another translation that is applied after the rotation.

Typically, this translation includes the inverse translation from the matrix  $S$  to reverse the translation for the pivot point for  $R$ .

To obtain the effective transformation at time  $t$ , the elements of the components of  $S$ ,  $R$ , and  $T$  will be interpolated linearly. The components are then multiplied to obtain the combined transformation  $C = T * R * S$ . The transformation  $C$  is the effective object-to-world transformations at time  $t$ , and  $C^{-1}$  is the effective world-to-object transformation at time  $t$ .

See also [OptixSRTMotionTransform::srtData](#), [optixConvertPointerToTraversableHandle\(\)](#)

### 7.51.2 Member Data Documentation

#### 7.51.2.1 a

`float OptixSRTData::a`

#### 7.51.2.2 b

`float OptixSRTData::b`

#### 7.51.2.3 c

`float OptixSRTData::c`

#### 7.51.2.4 pvx

`float OptixSRTData::pvx`

#### 7.51.2.5 pvy

`float OptixSRTData::pvy`

#### 7.51.2.6 pvz

`float OptixSRTData::pvz`

### 7.51.2.7 qw

float OptixSRTData::qw

### 7.51.2.8 qx

float OptixSRTData::qx

### 7.51.2.9 qy

float OptixSRTData::qy

### 7.51.2.10 qz

float OptixSRTData::qz

### 7.51.2.11 sx

float OptixSRTData::sx

### 7.51.2.12 sy

float OptixSRTData::sy

### 7.51.2.13 sz

float OptixSRTData::sz

### 7.51.2.14 tx

float OptixSRTData::tx

### 7.51.2.15 ty

float OptixSRTData::ty

### 7.51.2.16 tz

float OptixSRTData::tz

## 7.52 OptixSRTMotionTransform Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- [OptixTraversableHandle](#) child
- [OptixMotionOptions](#) motionOptions
- unsigned int pad [3]
- [OptixSRTData](#) srtData [2]

### 7.52.1 Detailed Description

Represents an SRT motion transformation.

The device address of instances of this type must be a multiple of OPTIX\_TRANSFORM\_BYTE\_ALIGNMENT.

This struct, as defined here, handles only N=2 motion keys due to the fixed array length of its `srtData` member. The following example shows how to create instances for an arbitrary number N of motion keys:

```
OptixSRTData srtData[N];
... // setup srtData
size_t transformSizeInBytes = sizeof(OptixSRTMotionTransform) + (N-2) * sizeof(OptixSRTData);
OptixSRTMotionTransform* srtMotionTransform = (OptixSRTMotionTransform*) malloc(transformSizeInBytes);
memset(srtMotionTransform, 0, transformSizeInBytes);
... // setup other members of srtMotionTransform
srtMotionTransform->motionOptions.numKeys = N;
memcpy(srtMotionTransform->srtData, srtData, N * sizeof(OptixSRTData));
... // copy srtMotionTransform to device memory
free(srtMotionTransform)
```

See also [optixConvertPointerToTraversableHandle\(\)](#)

## 7.52.2 Member Data Documentation

### 7.52.2.1 child

[OptixTraversableHandle](#) [OptixSRTMotionTransform::child](#)

The traversable transformed by this transformation.

### 7.52.2.2 motionOptions

[OptixMotionOptions](#) [OptixSRTMotionTransform::motionOptions](#)

The motion options for this transformation Must have at least two motion keys.

### 7.52.2.3 pad

`unsigned int` [OptixSRTMotionTransform::pad\[3\]](#)

Padding to make the SRT data 16 byte aligned.

### 7.52.2.4 srtData

[OptixSRTData](#) [OptixSRTMotionTransform::srtData\[2\]](#)

The actual SRT data describing the transformation.

## 7.53 OptixStackSize Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- `unsigned int` [cssRG](#)
- `unsigned int` [cssMS](#)
- `unsigned int` [cssCH](#)
- `unsigned int` [cssAH](#)
- `unsigned int` [cssIS](#)
- `unsigned int` [cssCC](#)
- `unsigned int` [dssDC](#)

### 7.53.1 Detailed Description

Describes the stack size requirements of a program group.

See also [optixProgramGroupGetStackSize\(\)](#)

## 7.53.2 Member Data Documentation

### 7.53.2.1 cssAH

`unsigned int OptixStackSizes::cssAH`

Continuation stack size of AH programs in bytes.

### 7.53.2.2 cssCC

`unsigned int OptixStackSizes::cssCC`

Continuation stack size of CC programs in bytes.

### 7.53.2.3 cssCH

`unsigned int OptixStackSizes::cssCH`

Continuation stack size of CH programs in bytes.

### 7.53.2.4 cssIS

`unsigned int OptixStackSizes::cssIS`

Continuation stack size of IS programs in bytes.

### 7.53.2.5 cssMS

`unsigned int OptixStackSizes::cssMS`

Continuation stack size of MS programs in bytes.

### 7.53.2.6 cssRG

`unsigned int OptixStackSizes::cssRG`

Continuation stack size of RG programs in bytes.

### 7.53.2.7 dssDC

`unsigned int OptixStackSizes::dssDC`

Direct stack size of DC programs in bytes.

## 7.54 OptixStaticTransform Struct Reference

```
#include <optix_types.h>
```

### Public Attributes

- [OptixTraversableHandle](#) child
- unsigned int pad [2]
- float transform [12]
- float invTransform [12]

### 7.54.1 Detailed Description

Static transform.

The device address of instances of this type must be a multiple of OPTIX\_TRANSFORM\_BYTE\_ALIGNMENT.



See also [optixConvertPointerToTraversableHandle\(\)](#)

## 7.54.2 Member Data Documentation

### 7.54.2.1 child

[OptixTraversableHandle](#) [OptixStaticTransform::child](#)

The traversable transformed by this transformation.

### 7.54.2.2 invTransform

float [OptixStaticTransform::invTransform\[12\]](#)

Affine world-to-object transformation as 3x4 matrix in row-major layout Must be the inverse of the transform matrix.

### 7.54.2.3 pad

unsigned int [OptixStaticTransform::pad\[2\]](#)

Padding to make the transformations 16 byte aligned.

### 7.54.2.4 transform

float [OptixStaticTransform::transform\[12\]](#)

Affine object-to-world transformation as 3x4 matrix in row-major layout.

## 7.55 OptixUtilDenoiserImageTile Struct Reference

`#include <optix_denoiser_tiling.h>`

### Public Attributes

- [OptixImage2D](#) input
- [OptixImage2D](#) output
- unsigned int [inputOffsetX](#)
- unsigned int [inputOffsetY](#)

### 7.55.1 Detailed Description

Tile definition.

see [optixUtilDenoiserSplitImage](#)

## 7.55.2 Member Data Documentation

### 7.55.2.1 input

[OptixImage2D](#) [OptixUtilDenoiserImageTile::input](#)

### 7.55.2.2 inputOffsetX

unsigned int [OptixUtilDenoiserImageTile::inputOffsetX](#)

### 7.55.2.3 inputOffsetY

unsigned int [OptixUtilDenoiserImageTile::inputOffsetY](#)

### 7.55.2.4 output

`OptixImage2D OptixUtilDenoiserImageTile::output`

## 7.56 optix\_internal::TypePack<... > Struct Template Reference

`#include <optix_device_impl.h>`

## 8 File Documentation

### 8.1 optix\_device\_impl.h File Reference

#### Classes

- struct `optix_internal::TypePack<... >`

#### Namespaces

- namespace `optix_internal`

#### Macros

- `#define OPTIX_DEFINE_optixGetAttribute_BODY(which)`
- `#define OPTIX_DEFINE_optixGetExceptionDetail_BODY(which)`

#### Functions

- `template<typename... Payload>`  
`static __forceinline__ __device__ void optixTrace (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBTOffset, unsigned int SBTStride, unsigned int missSBTIndex, Payload &... payload)`
- `template<typename... Payload>`  
`static __forceinline__ __device__ void optixTrace (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBTOffset, unsigned int SBTStride, unsigned int missSBTIndex, Payload &... payload)`
- `static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_5 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_6 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_7 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_8 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_9 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_10 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_11 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_12 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_13 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_14 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_15 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_16 (unsigned int p)`

- 
- NVIDIA OptiX 7.7 API

- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixGetWorldRayDirection ()
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixGetObjectRayOrigin ()
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixGetObjectRayDirection ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetRayTmin ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetRayTmax ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetRayTime ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetRayFlags ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetRayVisibilityMask ()
- static \_\_forceinline\_\_ \_\_device\_\_ OptixTraversableHandle optixGetInstanceTraversableFromIAS (OptixTraversableHandle ias, unsigned int instIdx)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetTriangleVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetMicroTriangleVertexData (float3 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetMicroTriangleBarycentricsData (float2 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetLinearCurveVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetQuadraticBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetCubicBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetCatmullRomVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetCubicBezierVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetRibbonVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixGetRibbonNormal (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetSphereData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static \_\_forceinline\_\_ \_\_device\_\_ OptixTraversableHandle optixGetGASTraversableHandle ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetGASMotionTimeBegin (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetGASMotionTimeEnd (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetGASMotionStepCount (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetWorldToObjectTransformMatrix (float m[12])
- static \_\_forceinline\_\_ \_\_device\_\_ void optixGetObjectToWorldTransformMatrix (float m[12])
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformPointFromWorldToObjectSpace (float3 point)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformVectorFromWorldToObjectSpace (float3 vec)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformNormalFromWorldToObjectSpace (float3 normal)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformPointFromObjectToWorldSpace (float3 point)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformVectorFromObjectToWorldSpace (float3 vec)

- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixTransformNormalFromObjectToWorldSpace (float3 normal)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetTransformListSize ()
- static \_\_forceinline\_\_ \_\_device\_\_ OptixTraversableHandle optixGetTransformListHandle (unsigned int index)
- static \_\_forceinline\_\_ \_\_device\_\_ OptixTransformType optixGetTransformTypeFromHandle (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ const OptixStaticTransform \* optixGetStaticTransformFromHandle (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ const OptixSRTMotionTransform \* optixGetSRTMotionTransformFromHandle (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ const OptixMatrixMotionTransform \* optixGetMatrixMotionTransformFromHandle (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetInstanceIdFromHandle (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ OptixTraversableHandle optixGetInstanceChildFromHandle (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ const float4 \* optixGetInstanceTransformFromHandle (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ const float4 \* optixGetInstanceInverseTransformFromHandle (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6, unsigned int a7)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_0 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_1 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_2 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_3 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_4 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_5 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_6 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_7 ()
- static \_\_forceinline\_\_ \_\_device\_\_ void optixTerminateRay ()
- static \_\_forceinline\_\_ \_\_device\_\_ void optixIgnoreIntersection ()

- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPrimitiveIndex](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetSbtGASIndex](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetInstanceId](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetInstanceIndex](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetHitKind](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ [OptixPrimitiveType](#) [optixGetPrimitiveType](#) (unsigned int hitKind)
- static \_\_forceinline\_\_ \_\_device\_\_ bool [optixIsBackFaceHit](#) (unsigned int hitKind)
- static \_\_forceinline\_\_ \_\_device\_\_ bool [optixIsFrontFaceHit](#) (unsigned int hitKind)
- static \_\_forceinline\_\_ \_\_device\_\_ [OptixPrimitiveType](#) [optixGetPrimitiveType](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool [optixIsBackFaceHit](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool [optixIsFrontFaceHit](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool [optixIsTriangleHit](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool [optixIsTriangleFrontFaceHit](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool [optixIsTriangleBackFaceHit](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool [optixIsDisplacedMicromeshTriangleHit](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool [optixIsDisplacedMicromeshTriangleFrontFaceHit](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool [optixIsDisplacedMicromeshTriangleBackFaceHit](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ float [optixGetCurveParameter](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ float2 [optixGetRibbonParameters](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ float2 [optixGetTriangleBarycentrics](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ uint3 [optixGetLaunchIndex](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ uint3 [optixGetLaunchDimensions](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ [CUdeviceptr](#) [optixGetSbtDataPointer](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixThrowException](#) (int exceptionCode)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixThrowException](#) (int exceptionCode, unsigned int exceptionDetail0)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixThrowException](#) (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixThrowException](#) (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixThrowException](#) (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixThrowException](#) (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixThrowException](#) (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixThrowException](#) (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixThrowException](#) (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6, unsigned int exceptionDetail7)
- static \_\_forceinline\_\_ \_\_device\_\_ int [optixGetExceptionCode](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetExceptionDetail\\_0](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetExceptionDetail\\_1](#) ()



- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetExceptionDetail\\_2](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetExceptionDetail\\_3](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetExceptionDetail\\_4](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetExceptionDetail\\_5](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetExceptionDetail\\_6](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetExceptionDetail\\_7](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ [OptixTraversableHandle](#) [optixGetExceptionInvalidTraversable](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ int [optixGetExceptionInvalidSbtOffset](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ [OptixInvalidRayExceptionDetails](#) [optixGetExceptionInvalidRay](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ [OptixParameterMismatchExceptionDetails](#) [optixGetExceptionParameterMismatch](#) ()
- static \_\_forceinline\_\_ \_\_device\_\_ char \* [optixGetExceptionLineInfo](#) ()
- template<typename ReturnT , typename... ArgTypes>  
static \_\_forceinline\_\_ \_\_device\_\_ ReturnT [optixDirectCall](#) (unsigned int sbtIndex, ArgTypes... args)
- template<typename ReturnT , typename... ArgTypes>  
static \_\_forceinline\_\_ \_\_device\_\_ ReturnT [optixContinuationCall](#) (unsigned int sbtIndex, ArgTypes... args)
- static \_\_forceinline\_\_ \_\_device\_\_ uint4 [optixTexFootprint2D](#) (unsigned long long tex, unsigned int texInfo, float x, float y, unsigned int \*singleMipLevel)
- static \_\_forceinline\_\_ \_\_device\_\_ uint4 [optixTexFootprint2DGrad](#) (unsigned long long tex, unsigned int texInfo, float x, float y, float dPdx\_x, float dPdx\_y, float dPdy\_x, float dPdy\_y, bool coarse, unsigned int \*singleMipLevel)
- static \_\_forceinline\_\_ \_\_device\_\_ uint4 [optixTexFootprint2DLod](#) (unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool coarse, unsigned int \*singleMipLevel)

## 8.1.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Device side implementation

## 8.1.2 Macro Definition Documentation

### 8.1.2.1 OPTIX\_DEFINE\_optixGetAttribute\_BODY

```
#define OPTIX_DEFINE_optixGetAttribute_BODY(  
    which )
```

Value:

```
    unsigned int ret;  
    \  
    asm("call (%0), _optix_get_attribute_" #which ", ();" : "=r"(ret) :);  
    \  
    return ret;
```

### 8.1.2.2 OPTIX\_DEFINE\_optixGetExceptionDetail\_BODY

```
#define OPTIX_DEFINE_optixGetExceptionDetail_BODY(  
    which )
```

Value:

```
    unsigned int ret;  
\  
    asm("call (%0), _optix_get_exception_detail_" #which ", ();" : "=r"(ret) :);  
\  
    return ret;
```

## 8.1.3 Function Documentation

### 8.1.3.1 optixContinuationCall()

```
template<typename ReturnT , typename... ArgTypes>  
static __forceinline__ __device__ ReturnT optixContinuationCall (  
    unsigned int sbtIndex,  
    ArgTypes... args ) [static]
```

### 8.1.3.2 optixDirectCall()

```
template<typename ReturnT , typename... ArgTypes>  
static __forceinline__ __device__ ReturnT optixDirectCall (  
    unsigned int sbtIndex,  
    ArgTypes... args ) [static]
```

### 8.1.3.3 optixGetAttribute\_0()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_0 ( ) [static]
```

### 8.1.3.4 optixGetAttribute\_1()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_1 ( ) [static]
```

### 8.1.3.5 optixGetAttribute\_2()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_2 ( ) [static]
```

### 8.1.3.6 optixGetAttribute\_3()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_3 ( ) [static]
```

### 8.1.3.7 optixGetAttribute\_4()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_4 ( ) [static]
```

### 8.1.3.8 optixGetAttribute\_5()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_5 ( ) [static]
```

### 8.1.3.9 optixGetAttribute\_6()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_6 ( ) [static]
```



### 8.1.3.10 optixGetAttribute\_7()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_7 ( ) [static]
```

### 8.1.3.11 optixGetCatmullRomVertexData()

```
static __forceinline__ __device__ void optixGetCatmullRomVertexData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
    float4 data[4] ) [static]
```

### 8.1.3.12 optixGetCubicBezierVertexData()

```
static __forceinline__ __device__ void optixGetCubicBezierVertexData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
    float4 data[4] ) [static]
```

### 8.1.3.13 optixGetCubicBSplineVertexData()

```
static __forceinline__ __device__ void optixGetCubicBSplineVertexData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
    float4 data[4] ) [static]
```

### 8.1.3.14 optixGetCurveParameter()

```
static __forceinline__ __device__ float optixGetCurveParameter ( ) [static]
```

### 8.1.3.15 optixGetExceptionCode()

```
static __forceinline__ __device__ int optixGetExceptionCode ( ) [static]
```

### 8.1.3.16 optixGetExceptionDetail\_0()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0 ( )
[static]
```

### 8.1.3.17 optixGetExceptionDetail\_1()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1 ( )
[static]
```

### 8.1.3.18 optixGetExceptionDetail\_2()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2 ( )
```

*[static]*

#### 8.1.3.19 optixGetExceptionDetail\_3()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3 ( )  
[static]
```

#### 8.1.3.20 optixGetExceptionDetail\_4()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4 ( )  
[static]
```

#### 8.1.3.21 optixGetExceptionDetail\_5()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5 ( )  
[static]
```

#### 8.1.3.22 optixGetExceptionDetail\_6()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6 ( )  
[static]
```

#### 8.1.3.23 optixGetExceptionDetail\_7()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7 ( )  
[static]
```

#### 8.1.3.24 optixGetExceptionInvalidRay()

```
static __forceinline__ __device__ OptixInvalidRayExceptionDetails  
optixGetExceptionInvalidRay ( ) [static]
```

#### 8.1.3.25 optixGetExceptionInvalidSbtOffset()

```
static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset ( )  
[static]
```

#### 8.1.3.26 optixGetExceptionInvalidTraversable()

```
static __forceinline__ __device__ OptixTraversableHandle  
optixGetExceptionInvalidTraversable ( ) [static]
```

#### 8.1.3.27 optixGetExceptionLineInfo()

```
static __forceinline__ __device__ char * optixGetExceptionLineInfo ( ) [static]
```

#### 8.1.3.28 optixGetExceptionParameterMismatch()

```
static __forceinline__ __device__ OptixParameterMismatchExceptionDetails  
optixGetExceptionParameterMismatch ( ) [static]
```

#### 8.1.3.29 optixGetGASMotionStepCount()

```
static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (   
    OptixTraversableHandle handle ) [static]
```

### 8.1.3.30 optixGetGASMotionTimeBegin()

```
static __forceinline__ __device__ float optixGetGASMotionTimeBegin (
    OptixTraversableHandle handle ) [static]
```

### 8.1.3.31 optixGetGASMotionTimeEnd()

```
static __forceinline__ __device__ float optixGetGASMotionTimeEnd (
    OptixTraversableHandle handle ) [static]
```

### 8.1.3.32 optixGetGASTraversableHandle()

```
static __forceinline__ __device__ OptixTraversableHandle
optixGetGASTraversableHandle ( ) [static]
```

### 8.1.3.33 optixGetHitKind()

```
static __forceinline__ __device__ unsigned int optixGetHitKind ( ) [static]
```

### 8.1.3.34 optixGetInstanceChildFromHandle()

```
static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle (
    OptixTraversableHandle handle ) [static]
```

### 8.1.3.35 optixGetInstanceId()

```
static __forceinline__ __device__ unsigned int optixGetInstanceId ( ) [static]
```

### 8.1.3.36 optixGetInstanceIdFromHandle()

```
static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle
(
    OptixTraversableHandle handle ) [static]
```

### 8.1.3.37 optixGetInstanceIndex()

```
static __forceinline__ __device__ unsigned int optixGetInstanceIndex ( )
[static]
```

### 8.1.3.38 optixGetInstanceInverseTransformFromHandle()

```
static __forceinline__ __device__ const float4 *
optixGetInstanceInverseTransformFromHandle (
    OptixTraversableHandle handle ) [static]
```

### 8.1.3.39 optixGetInstanceTransformFromHandle()

```
static __forceinline__ __device__ const float4 *
optixGetInstanceTransformFromHandle (
    OptixTraversableHandle handle ) [static]
```

## 8.1.3.40 optixGetInstanceTraversableFromIAS()

```
static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS (
    OptixTraversableHandle ias,
    unsigned int instIdx ) [static]
```

## 8.1.3.41 optixGetLaunchDimensions()

```
static __forceinline__ __device__ uint3 optixGetLaunchDimensions ( ) [static]
```

## 8.1.3.42 optixGetLaunchIndex()

```
static __forceinline__ __device__ uint3 optixGetLaunchIndex ( ) [static]
```

## 8.1.3.43 optixGetLinearCurveVertexData()

```
static __forceinline__ __device__ void optixGetLinearCurveVertexData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
    float4 data[2] ) [static]
```

## 8.1.3.44 optixGetMatrixMotionTransformFromHandle()

```
static __forceinline__ __device__ const OptixMatrixMotionTransform *
optixGetMatrixMotionTransformFromHandle (
    OptixTraversableHandle handle ) [static]
```

## 8.1.3.45 optixGetMicroTriangleBarycentricsData()

```
static __forceinline__ __device__ void optixGetMicroTriangleBarycentricsData
(
    float2 data[3] ) [static]
```

## 8.1.3.46 optixGetMicroTriangleVertexData()

```
static __forceinline__ __device__ void optixGetMicroTriangleVertexData (
    float3 data[3] ) [static]
```

## 8.1.3.47 optixGetObjectRayDirection()

```
static __forceinline__ __device__ float3 optixGetObjectRayDirection ( )
[static]
```

## 8.1.3.48 optixGetObjectRayOrigin()

```
static __forceinline__ __device__ float3 optixGetObjectRayOrigin ( ) [static]
```

#### 8.1.3.49 optixGetObjectToWorldTransformMatrix()

```
static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix  
(  
    float m[12] ) [static]
```

#### 8.1.3.50 optixGetPayload\_0()

```
static __forceinline__ __device__ unsigned int optixGetPayload_0 ( ) [static]
```

#### 8.1.3.51 optixGetPayload\_1()

```
static __forceinline__ __device__ unsigned int optixGetPayload_1 ( ) [static]
```

#### 8.1.3.52 optixGetPayload\_10()

```
static __forceinline__ __device__ unsigned int optixGetPayload_10 ( ) [static]
```

#### 8.1.3.53 optixGetPayload\_11()

```
static __forceinline__ __device__ unsigned int optixGetPayload_11 ( ) [static]
```

#### 8.1.3.54 optixGetPayload\_12()

```
static __forceinline__ __device__ unsigned int optixGetPayload_12 ( ) [static]
```

#### 8.1.3.55 optixGetPayload\_13()

```
static __forceinline__ __device__ unsigned int optixGetPayload_13 ( ) [static]
```

#### 8.1.3.56 optixGetPayload\_14()

```
static __forceinline__ __device__ unsigned int optixGetPayload_14 ( ) [static]
```

#### 8.1.3.57 optixGetPayload\_15()

```
static __forceinline__ __device__ unsigned int optixGetPayload_15 ( ) [static]
```

#### 8.1.3.58 optixGetPayload\_16()

```
static __forceinline__ __device__ unsigned int optixGetPayload_16 ( ) [static]
```

#### 8.1.3.59 optixGetPayload\_17()

```
static __forceinline__ __device__ unsigned int optixGetPayload_17 ( ) [static]
```

#### 8.1.3.60 optixGetPayload\_18()

```
static __forceinline__ __device__ unsigned int optixGetPayload_18 ( ) [static]
```

#### 8.1.3.61 optixGetPayload\_19()

```
static __forceinline__ __device__ unsigned int optixGetPayload_19 ( ) [static]
```

### 8.1.3.62 optixGetPayload\_2()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_2 ( ) *[static]*

### 8.1.3.63 optixGetPayload\_20()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_20 ( ) *[static]*

### 8.1.3.64 optixGetPayload\_21()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_21 ( ) *[static]*

### 8.1.3.65 optixGetPayload\_22()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_22 ( ) *[static]*

### 8.1.3.66 optixGetPayload\_23()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_23 ( ) *[static]*

### 8.1.3.67 optixGetPayload\_24()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_24 ( ) *[static]*

### 8.1.3.68 optixGetPayload\_25()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_25 ( ) *[static]*

### 8.1.3.69 optixGetPayload\_26()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_26 ( ) *[static]*

### 8.1.3.70 optixGetPayload\_27()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_27 ( ) *[static]*

### 8.1.3.71 optixGetPayload\_28()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_28 ( ) *[static]*

### 8.1.3.72 optixGetPayload\_29()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_29 ( ) *[static]*

### 8.1.3.73 optixGetPayload\_3()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_3 ( ) *[static]*

### 8.1.3.74 optixGetPayload\_30()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_30 ( ) *[static]*

### 8.1.3.75 optixGetPayload\_31()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_31 ( ) *[static]*

### 8.1.3.76 optixGetPayload\_4()

```
static __forceinline__ __device__ unsigned int optixGetPayload_4 ( ) [static]
```

### 8.1.3.77 optixGetPayload\_5()

```
static __forceinline__ __device__ unsigned int optixGetPayload_5 ( ) [static]
```

### 8.1.3.78 optixGetPayload\_6()

```
static __forceinline__ __device__ unsigned int optixGetPayload_6 ( ) [static]
```

### 8.1.3.79 optixGetPayload\_7()

```
static __forceinline__ __device__ unsigned int optixGetPayload_7 ( ) [static]
```

### 8.1.3.80 optixGetPayload\_8()

```
static __forceinline__ __device__ unsigned int optixGetPayload_8 ( ) [static]
```

### 8.1.3.81 optixGetPayload\_9()

```
static __forceinline__ __device__ unsigned int optixGetPayload_9 ( ) [static]
```

### 8.1.3.82 optixGetPrimitiveIndex()

```
static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ( )  
[static]
```

### 8.1.3.83 optixGetPrimitiveType() [1/2]

```
static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ( )  
[static]
```

### 8.1.3.84 optixGetPrimitiveType() [2/2]

```
static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ( )  
    unsigned int hitKind ) [static]
```

### 8.1.3.85 optixGetQuadraticBSplineVertexData()

```
static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData ( )  
    OptixTraversableHandle gas,  
    unsigned int primIdx,  
    unsigned int sbtGASIndex,  
    float time,  
    float4 data[3] ) [static]
```

### 8.1.3.86 optixGetRayFlags()

```
static __forceinline__ __device__ unsigned int optixGetRayFlags ( ) [static]
```

### 8.1.3.87 optixGetRayTime()

```
static __forceinline__ __device__ float optixGetRayTime ( ) [static]
```

## 8.1.3.88 optixGetRayTmax()

```
static __forceinline__ __device__ float optixGetRayTmax ( ) [static]
```

## 8.1.3.89 optixGetRayTmin()

```
static __forceinline__ __device__ float optixGetRayTmin ( ) [static]
```

## 8.1.3.90 optixGetRayVisibilityMask()

```
static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask ( )  
[static]
```

## 8.1.3.91 optixGetRibbonNormal()

```
static __forceinline__ __device__ float3 optixGetRibbonNormal (   
    OptixTraversableHandle gas,  
    unsigned int primIdx,  
    unsigned int sbtGASIndex,  
    float time,  
    float2 ribbonParameters ) [static]
```

## 8.1.3.92 optixGetRibbonParameters()

```
static __forceinline__ __device__ float2 optixGetRibbonParameters ( ) [static]
```

## 8.1.3.93 optixGetRibbonVertexData()

```
static __forceinline__ __device__ void optixGetRibbonVertexData (   
    OptixTraversableHandle gas,  
    unsigned int primIdx,  
    unsigned int sbtGASIndex,  
    float time,  
    float4 data[3] ) [static]
```

## 8.1.3.94 optixGetSbtDataPointer()

```
static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ( )  
[static]
```

## 8.1.3.95 optixGetSbtGASIndex()

```
static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ( ) [static]
```

## 8.1.3.96 optixGetSphereData()

```
static __forceinline__ __device__ void optixGetSphereData (   
    OptixTraversableHandle gas,  
    unsigned int primIdx,  
    unsigned int sbtGASIndex,  
    float time,  
    float4 data[1] ) [static]
```



### 8.1.3.97 optixGetSRTMotionTransformFromHandle()

```
static __forceinline__ __device__ const OptixSRTMotionTransform *  
optixGetSRTMotionTransformFromHandle (   
    OptixTraversableHandle handle ) [static]
```

### 8.1.3.98 optixGetStaticTransformFromHandle()

```
static __forceinline__ __device__ const OptixStaticTransform *  
optixGetStaticTransformFromHandle (   
    OptixTraversableHandle handle ) [static]
```

### 8.1.3.99 optixGetTransformListHandle()

```
static __forceinline__ __device__ OptixTraversableHandle  
optixGetTransformListHandle (   
    unsigned int index ) [static]
```

### 8.1.3.100 optixGetTransformListSize()

```
static __forceinline__ __device__ unsigned int optixGetTransformListSize ( )  
[static]
```

### 8.1.3.101 optixGetTransformTypeFromHandle()

```
static __forceinline__ __device__ OptixTransformType  
optixGetTransformTypeFromHandle (   
    OptixTraversableHandle handle ) [static]
```

### 8.1.3.102 optixGetTriangleBarycentrics()

```
static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ( )  
[static]
```

### 8.1.3.103 optixGetTriangleVertexData()

```
static __forceinline__ __device__ void optixGetTriangleVertexData (   
    OptixTraversableHandle gas,  
    unsigned int primIdx,  
    unsigned int sbtGASIndex,  
    float time,  
    float3 data[3] ) [static]
```

### 8.1.3.104 optixGetWorldRayDirection()

```
static __forceinline__ __device__ float3 optixGetWorldRayDirection ( ) [static]
```

### 8.1.3.105 optixGetWorldRayOrigin()

```
static __forceinline__ __device__ float3 optixGetWorldRayOrigin ( ) [static]
```

## 8.1.3.106 optixGetWorldToObjectTransformMatrix()

```
static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix
(
    float m[12] ) [static]
```

## 8.1.3.107 optixIgnoreIntersection()

```
static __forceinline__ __device__ void optixIgnoreIntersection ( ) [static]
```

## 8.1.3.108 optixIsBackFaceHit() [1/2]

```
static __forceinline__ __device__ bool optixIsBackFaceHit ( ) [static]
```

## 8.1.3.109 optixIsBackFaceHit() [2/2]

```
static __forceinline__ __device__ bool optixIsBackFaceHit (
    unsigned int hitKind ) [static]
```

## 8.1.3.110 optixIsDisplacedMicromeshTriangleBackFaceHit()

```
static __forceinline__ __device__ bool
optixIsDisplacedMicromeshTriangleBackFaceHit ( ) [static]
```

## 8.1.3.111 optixIsDisplacedMicromeshTriangleFrontFaceHit()

```
static __forceinline__ __device__ bool
optixIsDisplacedMicromeshTriangleFrontFaceHit ( ) [static]
```

## 8.1.3.112 optixIsDisplacedMicromeshTriangleHit()

```
static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit
( ) [static]
```

## 8.1.3.113 optixIsFrontFaceHit() [1/2]

```
static __forceinline__ __device__ bool optixIsFrontFaceHit ( ) [static]
```

## 8.1.3.114 optixIsFrontFaceHit() [2/2]

```
static __forceinline__ __device__ bool optixIsFrontFaceHit (
    unsigned int hitKind ) [static]
```

## 8.1.3.115 optixIsTriangleBackFaceHit()

```
static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ( ) [static]
```

## 8.1.3.116 optixIsTriangleFrontFaceHit()

```
static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ( ) [static]
```

## 8.1.3.117 optixIsTriangleHit()

```
static __forceinline__ __device__ bool optixIsTriangleHit ( ) [static]
```

## 8.1.3.118 optixReportIntersection() [1/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind ) [static]
```

## 8.1.3.119 optixReportIntersection() [2/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0 ) [static]
```

## 8.1.3.120 optixReportIntersection() [3/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1 ) [static]
```

## 8.1.3.121 optixReportIntersection() [4/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1,
    unsigned int a2 ) [static]
```

## 8.1.3.122 optixReportIntersection() [5/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1,
    unsigned int a2,
    unsigned int a3 ) [static]
```

## 8.1.3.123 optixReportIntersection() [6/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1,
    unsigned int a2,
```

```
    unsigned int a3,
    unsigned int a4 ) [static]
```

#### 8.1.3.124 optixReportIntersection() [7/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1,
    unsigned int a2,
    unsigned int a3,
    unsigned int a4,
    unsigned int a5 ) [static]
```

#### 8.1.3.125 optixReportIntersection() [8/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1,
    unsigned int a2,
    unsigned int a3,
    unsigned int a4,
    unsigned int a5,
    unsigned int a6 ) [static]
```

#### 8.1.3.126 optixReportIntersection() [9/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
    float hitT,
    unsigned int hitKind,
    unsigned int a0,
    unsigned int a1,
    unsigned int a2,
    unsigned int a3,
    unsigned int a4,
    unsigned int a5,
    unsigned int a6,
    unsigned int a7 ) [static]
```

#### 8.1.3.127 optixSetPayload\_0()

```
static __forceinline__ __device__ void optixSetPayload_0 (
    unsigned int p ) [static]
```

### 8.1.3.128 optixSetPayload\_1()

```
static __forceinline__ __device__ void optixSetPayload_1 (  
    unsigned int p ) [static]
```

### 8.1.3.129 optixSetPayload\_10()

```
static __forceinline__ __device__ void optixSetPayload_10 (  
    unsigned int p ) [static]
```

### 8.1.3.130 optixSetPayload\_11()

```
static __forceinline__ __device__ void optixSetPayload_11 (  
    unsigned int p ) [static]
```

### 8.1.3.131 optixSetPayload\_12()

```
static __forceinline__ __device__ void optixSetPayload_12 (  
    unsigned int p ) [static]
```

### 8.1.3.132 optixSetPayload\_13()

```
static __forceinline__ __device__ void optixSetPayload_13 (  
    unsigned int p ) [static]
```

### 8.1.3.133 optixSetPayload\_14()

```
static __forceinline__ __device__ void optixSetPayload_14 (  
    unsigned int p ) [static]
```

### 8.1.3.134 optixSetPayload\_15()

```
static __forceinline__ __device__ void optixSetPayload_15 (  
    unsigned int p ) [static]
```

### 8.1.3.135 optixSetPayload\_16()

```
static __forceinline__ __device__ void optixSetPayload_16 (  
    unsigned int p ) [static]
```

### 8.1.3.136 optixSetPayload\_17()

```
static __forceinline__ __device__ void optixSetPayload_17 (  
    unsigned int p ) [static]
```

### 8.1.3.137 optixSetPayload\_18()

```
static __forceinline__ __device__ void optixSetPayload_18 (  
    unsigned int p ) [static]
```

### 8.1.3.138 optixSetPayload\_19()

```
static __forceinline__ __device__ void optixSetPayload_19 (  

```

unsigned int *p* ) *[static]*

#### 8.1.3.139 optixSetPayload\_2()

```
static __forceinline__ __device__ void optixSetPayload_2 (  
    unsigned int p ) [static]
```

#### 8.1.3.140 optixSetPayload\_20()

```
static __forceinline__ __device__ void optixSetPayload_20 (  
    unsigned int p ) [static]
```

#### 8.1.3.141 optixSetPayload\_21()

```
static __forceinline__ __device__ void optixSetPayload_21 (  
    unsigned int p ) [static]
```

#### 8.1.3.142 optixSetPayload\_22()

```
static __forceinline__ __device__ void optixSetPayload_22 (  
    unsigned int p ) [static]
```

#### 8.1.3.143 optixSetPayload\_23()

```
static __forceinline__ __device__ void optixSetPayload_23 (  
    unsigned int p ) [static]
```

#### 8.1.3.144 optixSetPayload\_24()

```
static __forceinline__ __device__ void optixSetPayload_24 (  
    unsigned int p ) [static]
```

#### 8.1.3.145 optixSetPayload\_25()

```
static __forceinline__ __device__ void optixSetPayload_25 (  
    unsigned int p ) [static]
```

#### 8.1.3.146 optixSetPayload\_26()

```
static __forceinline__ __device__ void optixSetPayload_26 (  
    unsigned int p ) [static]
```

#### 8.1.3.147 optixSetPayload\_27()

```
static __forceinline__ __device__ void optixSetPayload_27 (  
    unsigned int p ) [static]
```

#### 8.1.3.148 optixSetPayload\_28()

```
static __forceinline__ __device__ void optixSetPayload_28 (  
    unsigned int p ) [static]
```

#### 8.1.3.149 optixSetPayload\_29()

```
static __forceinline__ __device__ void optixSetPayload_29 (  
    unsigned int p ) [static]
```

#### 8.1.3.150 optixSetPayload\_3()

```
static __forceinline__ __device__ void optixSetPayload_3 (  
    unsigned int p ) [static]
```

#### 8.1.3.151 optixSetPayload\_30()

```
static __forceinline__ __device__ void optixSetPayload_30 (  
    unsigned int p ) [static]
```

#### 8.1.3.152 optixSetPayload\_31()

```
static __forceinline__ __device__ void optixSetPayload_31 (  
    unsigned int p ) [static]
```

#### 8.1.3.153 optixSetPayload\_4()

```
static __forceinline__ __device__ void optixSetPayload_4 (  
    unsigned int p ) [static]
```

#### 8.1.3.154 optixSetPayload\_5()

```
static __forceinline__ __device__ void optixSetPayload_5 (  
    unsigned int p ) [static]
```

#### 8.1.3.155 optixSetPayload\_6()

```
static __forceinline__ __device__ void optixSetPayload_6 (  
    unsigned int p ) [static]
```

#### 8.1.3.156 optixSetPayload\_7()

```
static __forceinline__ __device__ void optixSetPayload_7 (  
    unsigned int p ) [static]
```

#### 8.1.3.157 optixSetPayload\_8()

```
static __forceinline__ __device__ void optixSetPayload_8 (  
    unsigned int p ) [static]
```

#### 8.1.3.158 optixSetPayload\_9()

```
static __forceinline__ __device__ void optixSetPayload_9 (  
    unsigned int p ) [static]
```

#### 8.1.3.159 optixSetPayloadTypes()

```
static __forceinline__ __device__ void optixSetPayloadTypes (  

```

```
    unsigned int types ) [static]
```

#### 8.1.3.160 optixTerminateRay()

```
static __forceinline__ __device__ void optixTerminateRay ( ) [static]
```

#### 8.1.3.161 optixTexFootprint2D()

```
static __forceinline__ __device__ uint4 optixTexFootprint2D (
    unsigned long long tex,
    unsigned int texInfo,
    float x,
    float y,
    unsigned int * singleMipLevel ) [static]
```

#### 8.1.3.162 optixTexFootprint2DGrad()

```
static __forceinline__ __device__ uint4 optixTexFootprint2DGrad (
    unsigned long long tex,
    unsigned int texInfo,
    float x,
    float y,
    float dPdx_x,
    float dPdx_y,
    float dPdy_x,
    float dPdy_y,
    bool coarse,
    unsigned int * singleMipLevel ) [static]
```

#### 8.1.3.163 optixTexFootprint2DLod()

```
static __forceinline__ __device__ uint4 optixTexFootprint2DLod (
    unsigned long long tex,
    unsigned int texInfo,
    float x,
    float y,
    float level,
    bool coarse,
    unsigned int * singleMipLevel ) [static]
```

#### 8.1.3.164 optixThrowException() [1/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode ) [static]
```

#### 8.1.3.165 optixThrowException() [2/9]

```
static __forceinline__ __device__ void optixThrowException (
```



```
    int exceptionCode,  
    unsigned int exceptionDetail0 ) [static]
```

#### 8.1.3.166 optixThrowException() [3/9]

```
static __forceinline__ __device__ void optixThrowException (  
    int exceptionCode,  
    unsigned int exceptionDetail0,  
    unsigned int exceptionDetail1 ) [static]
```

#### 8.1.3.167 optixThrowException() [4/9]

```
static __forceinline__ __device__ void optixThrowException (  
    int exceptionCode,  
    unsigned int exceptionDetail0,  
    unsigned int exceptionDetail1,  
    unsigned int exceptionDetail2 ) [static]
```

#### 8.1.3.168 optixThrowException() [5/9]

```
static __forceinline__ __device__ void optixThrowException (  
    int exceptionCode,  
    unsigned int exceptionDetail0,  
    unsigned int exceptionDetail1,  
    unsigned int exceptionDetail2,  
    unsigned int exceptionDetail3 ) [static]
```

#### 8.1.3.169 optixThrowException() [6/9]

```
static __forceinline__ __device__ void optixThrowException (  
    int exceptionCode,  
    unsigned int exceptionDetail0,  
    unsigned int exceptionDetail1,  
    unsigned int exceptionDetail2,  
    unsigned int exceptionDetail3,  
    unsigned int exceptionDetail4 ) [static]
```

#### 8.1.3.170 optixThrowException() [7/9]

```
static __forceinline__ __device__ void optixThrowException (  
    int exceptionCode,  
    unsigned int exceptionDetail0,  
    unsigned int exceptionDetail1,  
    unsigned int exceptionDetail2,  
    unsigned int exceptionDetail3,  
    unsigned int exceptionDetail4,  
    unsigned int exceptionDetail5 ) [static]
```

## 8.1.3.171 optixThrowException() [8/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode,
    unsigned int exceptionDetail0,
    unsigned int exceptionDetail1,
    unsigned int exceptionDetail2,
    unsigned int exceptionDetail3,
    unsigned int exceptionDetail4,
    unsigned int exceptionDetail5,
    unsigned int exceptionDetail6 ) [static]
```

## 8.1.3.172 optixThrowException() [9/9]

```
static __forceinline__ __device__ void optixThrowException (
    int exceptionCode,
    unsigned int exceptionDetail0,
    unsigned int exceptionDetail1,
    unsigned int exceptionDetail2,
    unsigned int exceptionDetail3,
    unsigned int exceptionDetail4,
    unsigned int exceptionDetail5,
    unsigned int exceptionDetail6,
    unsigned int exceptionDetail7 ) [static]
```

## 8.1.3.173 optixTrace() [1/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixTrace (
    OptixPayloadTypeID type,
    OptixTraversableHandle handle,
    float3 rayOrigin,
    float3 rayDirection,
    float tmin,
    float tmax,
    float rayTime,
    OptixVisibilityMask visibilityMask,
    unsigned int rayFlags,
    unsigned int SBToffset,
    unsigned int SBTstride,
    unsigned int missSBTIndex,
    Payload &... payload ) [static]
```

## 8.1.3.174 optixTrace() [2/2]

```
template<typename... Payload>
```

```
static __forceinline__ __device__ void optixTrace (
    OptixTraversableHandle handle,
    float3 rayOrigin,
    float3 rayDirection,
    float tmin,
    float tmax,
    float rayTime,
    OptixVisibilityMask visibilityMask,
    unsigned int rayFlags,
    unsigned int SBToffset,
    unsigned int SBTstride,
    unsigned int missSBTIndex,
    Payload &... payload ) [static]
```

#### 8.1.3.175 optixTransformNormalFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixTransformNormalFromObjectToWorldSpace (
    float3 normal ) [static]
```

#### 8.1.3.176 optixTransformNormalFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixTransformNormalFromWorldToObjectSpace (
    float3 normal ) [static]
```

#### 8.1.3.177 optixTransformPointFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixTransformPointFromObjectToWorldSpace (
    float3 point ) [static]
```

#### 8.1.3.178 optixTransformPointFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixTransformPointFromWorldToObjectSpace (
    float3 point ) [static]
```

#### 8.1.3.179 optixTransformVectorFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixTransformVectorFromObjectToWorldSpace (
    float3 vec ) [static]
```

#### 8.1.3.180 optixTransformVectorFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixTransformVectorFromWorldToObjectSpace (
```

```
float3 vec ) [static]
```

### 8.1.3.181 optixUndefinedValue()

```
static __forceinline__ __device__ unsigned int optixUndefinedValue ( ) [static]
```

## 8.2 optix\_device\_impl.h

[Go to the documentation of this file.](#)

```
1 /*
2  * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
3  *
4  * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5  * rights in and to this software, related documentation and any modifications thereto.
6  * Any use, reproduction, disclosure or distribution of this software and related
7  * documentation without an express license agreement from NVIDIA Corporation is strictly
8  * prohibited.
9  *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE. IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
21 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
22 #error("optix_device_impl.h is an internal header file and must not be used directly. Please use
23 optix_device.h or optix.h instead.")
24 #endif
25
26 #ifndef OPTIX_DEVICE_IMPL_H
27 #define OPTIX_DEVICE_IMPL_H
28
29 #include "internal/optix_device_impl_exception.h"
30 #include "internal/optix_device_impl_transformations.h"
31
32 #ifndef __CUDACC_RTC__
33 #include <initializer_list>
34 #include <type_traits>
35 #endif
36
37 namespace optix_internal {
38 template <typename...>
39 struct TypePack{};
40 } // namespace optix_internal
41
42 template <typename... Payload>
43 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
44 float3 rayOrigin,
45 float3 rayDirection,
46 float tmin,
47 float tmax,
48 float rayTime,
49 OptixVisibilityMask visibilityMask,
50 unsigned int rayFlags,
51 unsigned int SBTOffset,
52 unsigned int SBTStride,
53 unsigned int missSBTIndex,
54 Payload&... payload)
55 {
56     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
57 }
```

```

64 // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
65 // TypePack 1 unsigned int T0 T1 T2 ... Tn-1 Tn
66 // TypePack 2 T0 T1 T2 T3 ... Tn unsigned int
67 #ifndef __CUDA_RTC__
68 static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int>::value,
69 "All payload parameters need to be unsigned int.");
70 #endif
71
72 OptixPayloadTypeID type = OPTIX_PAYLOAD_TYPE_DEFAULT;
73 float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
74 float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
75 unsigned int p[33] = { 0, payload... };
76 int payloadSize = (int)sizeof...(Payload);
77 asm volatile(
78 "call"
79
80 "(%0,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18,%19,%20,%21,%22,%23,%24,%25,%26,%27,%28,%29,%30,%31),"
81 "_optix_trace_typed_32,"
82
83 "(%32,%33,%34,%35,%36,%37,%38,%39,%40,%41,%42,%43,%44,%45,%46,%47,%48,%49,%50,%51,%52,%53,%54,%55,%56,%57,%58,%59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
84 : "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]), "r"(p[6]), "r"(p[7]),
85 "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]), "r"(p[14]),
86 "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]), "r"(p[21]),
87 "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]), "r"(p[28]),
88 "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
89 : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
90 "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBToffset), "r"(SBTstride),
91 "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
92 "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
93 "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
94 "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
95 "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
96 :);
97 unsigned int index = 1;
98 (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
99 }
100
101
102 template <typename... Payload>
103 static __forceinline__ __device__ void optixTrace(OptixPayloadTypeID type,
104 OptixTraversableHandle handle,
105 float3 rayOrigin,
106 float3 rayDirection,
107 float tmin,
108 float tmax,
109 float rayTime,
110 OptixVisibilityMask visibilityMask,
111 unsigned int rayFlags,
112 unsigned int SBToffset,
113 unsigned int SBTstride,
114 unsigned int missSBTIndex,
115 Payload&... payload)
116 {
117 // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
118 // TypePack 1 unsigned int T0 T1 T2 ... Tn-1 Tn
119 // TypePack 2 T0 T1 T2 T3 ... Tn unsigned int
120 static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
121 #ifndef __CUDA_RTC__
122 static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int>::value,
123 "All payload parameters need to be unsigned int.");
124 #endif
125
126 float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;

```

```

127     float      dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
128     unsigned int p[33]      = {0, payload...};
129     int         payloadSize = (int)sizeof...(Payload);
130
131     asm volatile(
132         "call"
133
134         "(%0,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18,%19,%20,%21,%22,%23,%24,%25,%26,%27,%28,%29,%30,%31),"
135         "_optix_trace_typed_32,"
136
137         "(%32,%33,%34,%35,%36,%37,%38,%39,%40,%41,%42,%43,%44,%45,%46,%47,%48,%49,%50,%51,%52,%53,%54,%55,%56,%57,%58,%59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
138         : "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]), "r"(p[6]), "r"(p[7]),
139           "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]), "r"(p[14]),
140           "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]), "r"(p[21]),
141           "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]), "r"(p[28]),
142           "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
143         : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
144           "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBTOffset), "r"(SBTstride),
145           "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
146           "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
147           "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
148           "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
149           "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
150         :);
151     unsigned int index = 1;
152     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
153 }
154
155
156 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p)
157 {
158     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(0), "r"(p) :);
159 }
160
161 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p)
162 {
163     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(1), "r"(p) :);
164 }
165
166 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p)
167 {
168     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(2), "r"(p) :);
169 }
170
171 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p)
172 {
173     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(3), "r"(p) :);
174 }
175
176 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p)
177 {
178     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(4), "r"(p) :);
179 }
180
181 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p)
182 {
183     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(5), "r"(p) :);
184 }
185
186 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p)
187 {
188     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(6), "r"(p) :);
189 }
190
191 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p)

```

```

192 {
193     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(7), "r"(p) :);
194 }
195
196 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p)
197 {
198     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(8), "r"(p) :);
199 }
200
201 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p)
202 {
203     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(9), "r"(p) :);
204 }
205
206 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p)
207 {
208     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(10), "r"(p) :);
209 }
210
211 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p)
212 {
213     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(11), "r"(p) :);
214 }
215
216 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p)
217 {
218     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(12), "r"(p) :);
219 }
220
221 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p)
222 {
223     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(13), "r"(p) :);
224 }
225
226 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p)
227 {
228     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(14), "r"(p) :);
229 }
230
231 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p)
232 {
233     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(15), "r"(p) :);
234 }
235
236 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p)
237 {
238     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(16), "r"(p) :);
239 }
240
241 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p)
242 {
243     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(17), "r"(p) :);
244 }
245
246 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p)
247 {
248     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(18), "r"(p) :);
249 }
250
251 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p)
252 {
253     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(19), "r"(p) :);
254 }
255
256 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p)
257 {
258     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(20), "r"(p) :);

```

```

259 }
260
261 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p)
262 {
263     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(21), "r"(p) :);
264 }
265
266 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p)
267 {
268     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(22), "r"(p) :);
269 }
270
271 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p)
272 {
273     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(23), "r"(p) :);
274 }
275
276 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p)
277 {
278     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(24), "r"(p) :);
279 }
280
281 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p)
282 {
283     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(25), "r"(p) :);
284 }
285
286 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p)
287 {
288     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(26), "r"(p) :);
289 }
290
291 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p)
292 {
293     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(27), "r"(p) :);
294 }
295
296 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p)
297 {
298     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(28), "r"(p) :);
299 }
300
301 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p)
302 {
303     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(29), "r"(p) :);
304 }
305
306 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p)
307 {
308     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(30), "r"(p) :);
309 }
310
311 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p)
312 {
313     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(31), "r"(p) :);
314 }
315
316 static __forceinline__ __device__ unsigned int optixGetPayload_0()
317 {
318     unsigned int result;
319     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(0) :);
320     return result;
321 }
322
323 static __forceinline__ __device__ unsigned int optixGetPayload_1()
324 {
325     unsigned int result;

```



```

326     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(1) :);
327     return result;
328 }
329
330 static __forceinline__ __device__ unsigned int optixGetPayload_2()
331 {
332     unsigned int result;
333     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(2) :);
334     return result;
335 }
336
337 static __forceinline__ __device__ unsigned int optixGetPayload_3()
338 {
339     unsigned int result;
340     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(3) :);
341     return result;
342 }
343
344 static __forceinline__ __device__ unsigned int optixGetPayload_4()
345 {
346     unsigned int result;
347     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(4) :);
348     return result;
349 }
350
351 static __forceinline__ __device__ unsigned int optixGetPayload_5()
352 {
353     unsigned int result;
354     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(5) :);
355     return result;
356 }
357
358 static __forceinline__ __device__ unsigned int optixGetPayload_6()
359 {
360     unsigned int result;
361     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(6) :);
362     return result;
363 }
364
365 static __forceinline__ __device__ unsigned int optixGetPayload_7()
366 {
367     unsigned int result;
368     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(7) :);
369     return result;
370 }
371
372 static __forceinline__ __device__ unsigned int optixGetPayload_8()
373 {
374     unsigned int result;
375     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(8) :);
376     return result;
377 }
378
379 static __forceinline__ __device__ unsigned int optixGetPayload_9()
380 {
381     unsigned int result;
382     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(9) :);
383     return result;
384 }
385
386 static __forceinline__ __device__ unsigned int optixGetPayload_10()
387 {
388     unsigned int result;
389     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(10) :);
390     return result;
391 }
392

```

```

393 static __forceinline__ __device__ unsigned int optixGetPayload_11()
394 {
395     unsigned int result;
396     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(11) :);
397     return result;
398 }
399
400 static __forceinline__ __device__ unsigned int optixGetPayload_12()
401 {
402     unsigned int result;
403     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(12) :);
404     return result;
405 }
406
407 static __forceinline__ __device__ unsigned int optixGetPayload_13()
408 {
409     unsigned int result;
410     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(13) :);
411     return result;
412 }
413
414 static __forceinline__ __device__ unsigned int optixGetPayload_14()
415 {
416     unsigned int result;
417     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(14) :);
418     return result;
419 }
420
421 static __forceinline__ __device__ unsigned int optixGetPayload_15()
422 {
423     unsigned int result;
424     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(15) :);
425     return result;
426 }
427
428 static __forceinline__ __device__ unsigned int optixGetPayload_16()
429 {
430     unsigned int result;
431     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(16) :);
432     return result;
433 }
434
435 static __forceinline__ __device__ unsigned int optixGetPayload_17()
436 {
437     unsigned int result;
438     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(17) :);
439     return result;
440 }
441
442 static __forceinline__ __device__ unsigned int optixGetPayload_18()
443 {
444     unsigned int result;
445     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(18) :);
446     return result;
447 }
448
449 static __forceinline__ __device__ unsigned int optixGetPayload_19()
450 {
451     unsigned int result;
452     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(19) :);
453     return result;
454 }
455
456 static __forceinline__ __device__ unsigned int optixGetPayload_20()
457 {
458     unsigned int result;
459     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(20) :);

```

```

460     return result;
461 }
462
463 static __forceinline__ __device__ unsigned int optixGetPayload_21()
464 {
465     unsigned int result;
466     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(21) :);
467     return result;
468 }
469
470 static __forceinline__ __device__ unsigned int optixGetPayload_22()
471 {
472     unsigned int result;
473     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(22) :);
474     return result;
475 }
476
477 static __forceinline__ __device__ unsigned int optixGetPayload_23()
478 {
479     unsigned int result;
480     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(23) :);
481     return result;
482 }
483
484 static __forceinline__ __device__ unsigned int optixGetPayload_24()
485 {
486     unsigned int result;
487     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(24) :);
488     return result;
489 }
490
491 static __forceinline__ __device__ unsigned int optixGetPayload_25()
492 {
493     unsigned int result;
494     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(25) :);
495     return result;
496 }
497
498 static __forceinline__ __device__ unsigned int optixGetPayload_26()
499 {
500     unsigned int result;
501     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(26) :);
502     return result;
503 }
504
505 static __forceinline__ __device__ unsigned int optixGetPayload_27()
506 {
507     unsigned int result;
508     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(27) :);
509     return result;
510 }
511
512 static __forceinline__ __device__ unsigned int optixGetPayload_28()
513 {
514     unsigned int result;
515     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(28) :);
516     return result;
517 }
518
519 static __forceinline__ __device__ unsigned int optixGetPayload_29()
520 {
521     unsigned int result;
522     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(29) :);
523     return result;
524 }
525
526 static __forceinline__ __device__ unsigned int optixGetPayload_30()

```

```

527 {
528     unsigned int result;
529     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(30) :);
530     return result;
531 }
532
533 static __forceinline__ __device__ unsigned int optixGetPayload_31()
534 {
535     unsigned int result;
536     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(31) :);
537     return result;
538 }
539
540 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int types)
541 {
542     asm volatile("call _optix_set_payload_types, (%0);" : : "r"(types) :);
543 }
544
545 static __forceinline__ __device__ unsigned int optixUndefinedValue()
546 {
547     unsigned int u0;
548     asm("call (%0), _optix_undef_value, ();" : "=r"(u0) :);
549     return u0;
550 }
551
552 static __forceinline__ __device__ float3 optixGetWorldRayOrigin()
553 {
554     float f0, f1, f2;
555     asm("call (%0), _optix_get_world_ray_origin_x, ();" : "=f"(f0) :);
556     asm("call (%0), _optix_get_world_ray_origin_y, ();" : "=f"(f1) :);
557     asm("call (%0), _optix_get_world_ray_origin_z, ();" : "=f"(f2) :);
558     return make_float3(f0, f1, f2);
559 }
560
561 static __forceinline__ __device__ float3 optixGetWorldRayDirection()
562 {
563     float f0, f1, f2;
564     asm("call (%0), _optix_get_world_ray_direction_x, ();" : "=f"(f0) :);
565     asm("call (%0), _optix_get_world_ray_direction_y, ();" : "=f"(f1) :);
566     asm("call (%0), _optix_get_world_ray_direction_z, ();" : "=f"(f2) :);
567     return make_float3(f0, f1, f2);
568 }
569
570 static __forceinline__ __device__ float3 optixGetObjectRayOrigin()
571 {
572     float f0, f1, f2;
573     asm("call (%0), _optix_get_object_ray_origin_x, ();" : "=f"(f0) :);
574     asm("call (%0), _optix_get_object_ray_origin_y, ();" : "=f"(f1) :);
575     asm("call (%0), _optix_get_object_ray_origin_z, ();" : "=f"(f2) :);
576     return make_float3(f0, f1, f2);
577 }
578
579 static __forceinline__ __device__ float3 optixGetObjectRayDirection()
580 {
581     float f0, f1, f2;
582     asm("call (%0), _optix_get_object_ray_direction_x, ();" : "=f"(f0) :);
583     asm("call (%0), _optix_get_object_ray_direction_y, ();" : "=f"(f1) :);
584     asm("call (%0), _optix_get_object_ray_direction_z, ();" : "=f"(f2) :);
585     return make_float3(f0, f1, f2);
586 }
587
588 static __forceinline__ __device__ float optixGetRayTmin()
589 {
590     float f0;
591     asm("call (%0), _optix_get_ray_tmin, ();" : "=f"(f0) :);
592     return f0;
593 }

```

```

594
595 static __forceinline__ __device__ float optixGetRayTmax()
596 {
597     float f0;
598     asm("call (%0), _optix_get_ray_tmax, ();" : "=f"(f0) :);
599     return f0;
600 }
601
602 static __forceinline__ __device__ float optixGetRayTime()
603 {
604     float f0;
605     asm("call (%0), _optix_get_ray_time, ();" : "=f"(f0) :);
606     return f0;
607 }
608
609 static __forceinline__ __device__ unsigned int optixGetRayFlags()
610 {
611     unsigned int u0;
612     asm("call (%0), _optix_get_ray_flags, ();" : "=r"(u0) :);
613     return u0;
614 }
615
616 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask()
617 {
618     unsigned int u0;
619     asm("call (%0), _optix_get_ray_visibility_mask, ();" : "=r"(u0) :);
620     return u0;
621 }
622
623 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias,
624                                     unsigned int
instIdx)
625 {
626     unsigned long long handle;
627     asm("call (%0), _optix_get_instance_traversable_from_ias, (%1, %2);"
        : "=l"(handle) : "l"(ias), "r"(instIdx));
628     return (OptixTraversableHandle)handle;
629 }
630
631
632
633 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas,
634   unsigned int      primIdx,
635   unsigned int      sbtGASIndex,
636   float              time,
637   float3             data[3])
638 {
639     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_triangle_vertex_data, "
        "(%9, %10, %11, %12);"
        : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
        "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
        : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
        :);
640 }
641
642
643
644
645
646
647 static __forceinline__ __device__ void optixGetMicroTriangleVertexData(float3 data[3])
648 {
649     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_microtriangle_vertex_data, "
        "(%9, %10, %11, %12);"
        : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
        "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
        : : :);
650 }
651
652
653
654
655 static __forceinline__ __device__ void optixGetMicroTriangleBarycentricsData(float2 data[3])
656 {
657     asm("call (%0, %1, %2, %3, %4, %5), _optix_get_microtriangle_barycentrics_data, "
        "(%6, %7, %8, %9, %10, %11);"
658     : : :);

```

```

659         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[1].x), "=f"(data[1].y), "=f"(data[2].x),
        "=f"(data[2].y)
660         :);
661     }
662
663     static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
664                                     unsigned int primIdx,
665                                     unsigned int sbtGASIndex,
666                                     float time,
667                                     float4 data[2])
668     {
669         asm("call (%0, %1, %2, %3, %4, %5, %6, %7), _optix_get_linear_curve_vertex_data, "
670             "(%8, %9, %10, %11);"
671             : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
672               "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
673             : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
674             :);
675     }
676
677     static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(OptixTraversableHandle gas,
678                                     unsigned int primIdx,
679                                     unsigned int sbtGASIndex,
680                                     float time,
681                                     float4 data[3])
682     {
683         asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),
684             _optix_get_quadratic_bspline_vertex_data, "
685             "(%12, %13, %14, %15);"
686             : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
687               "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
688               "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w)
689             : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
690             :);
691     }
692
693     static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,
694                                     unsigned int primIdx,
695                                     unsigned int sbtGASIndex,
696                                     float time,
697                                     float4 data[4])
698     {
699         asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
700             "_optix_get_cubic_bspline_vertex_data, "
701             "(%16, %17, %18, %19);"
702             : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
703               "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
704               "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
705               "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
706             : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
707             :);
708     }
709
710     static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas,
711                                     unsigned int primIdx,
712                                     unsigned int sbtGASIndex,
713                                     float time,
714                                     float4 data[4])
715     {
716         asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
717             "_optix_get_catmullrom_vertex_data, "
718             "(%16, %17, %18, %19);"
719             : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
720               "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
721               "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
722               "=f"(data[3].w)
723             : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
724             :);

```

```

723 }
724
725 static __forceinline__ __device__ void optixGetCubicBezierVertexData(OptixTraversableHandle gas,
726                               unsigned int      primIdx,
727                               unsigned int      sbtGASIndex,
728                               float            time,
729                               float4          data[4])
730 {
731     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
732         "_optix_get_cubic_bezier_vertex_data, "
733         "(%16, %17, %18, %19);"
734         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
735         "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
736         "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
737         "=f"(data[3].w)
738         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
739     );
740
741 static __forceinline__ __device__ void optixGetRibbonVertexData(OptixTraversableHandle gas,
742                               unsigned int      primIdx,
743                               unsigned int      sbtGASIndex,
744                               float            time,
745                               float4          data[3])
746 {
747     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11), _optix_get_ribbon_vertex_data, "
748         "(%12, %13, %14, %15);"
749         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
750         "=f"(data[1].y),
751         "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
752         "=f"(data[2].w)
753         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
754     );
755
756 static __forceinline__ __device__ float3 optixGetRibbonNormal(OptixTraversableHandle gas,
757                               unsigned int      primIdx,
758                               unsigned int      sbtGASIndex,
759                               float            time,
760                               float2          ribbonParameters)
761 {
762     float3 normal;
763     asm("call (%0, %1, %2), _optix_get_ribbon_normal, "
764         "(%3, %4, %5, %6, %7, %8);"
765         : "=f"(normal.x), "=f"(normal.y), "=f"(normal.z)
766         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time),
767         "f"(ribbonParameters.x), "f"(ribbonParameters.y)
768     );
769     return normal;
770 }
771
772 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas,
773                               unsigned int      primIdx,
774                               unsigned int      sbtGASIndex,
775                               float            time,
776                               float4          data[1])
777 {
778     asm("call (%0, %1, %2, %3), "
779         "_optix_get_sphere_data, "
780         "(%4, %5, %6, %7);"
781         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
782         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
783     );
784
785 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle()
786 {

```

```

787     unsigned long long handle;
788     asm("call (%0), _optix_get_gas_traversable_handle, ("); : "=l"(handle) :);
789     return (OptixTraversableHandle)handle;
790 }
791
792 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle handle)
793 {
794     float f0;
795     asm("call (%0), _optix_get_gas_motion_time_begin, (%1);" : "=f"(f0) : "l"(handle) :);
796     return f0;
797 }
798
799 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle handle)
800 {
801     float f0;
802     asm("call (%0), _optix_get_gas_motion_time_end, (%1);" : "=f"(f0) : "l"(handle) :);
803     return f0;
804 }
805
806 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle handle)
807 {
808     unsigned int u0;
809     asm("call (%0), _optix_get_gas_motion_step_count, (%1);" : "=r"(u0) : "l"(handle) :);
810     return u0;
811 }
812
813 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12])
814 {
815     if(optixGetTransformListSize() == 0)
816     {
817         m[0] = 1.0f;
818         m[1] = 0.0f;
819         m[2] = 0.0f;
820         m[3] = 0.0f;
821         m[4] = 0.0f;
822         m[5] = 1.0f;
823         m[6] = 0.0f;
824         m[7] = 0.0f;
825         m[8] = 0.0f;
826         m[9] = 0.0f;
827         m[10] = 1.0f;
828         m[11] = 0.0f;
829         return;
830     }
831
832     float4 m0, m1, m2;
833     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
834     m[0] = m0.x;
835     m[1] = m0.y;
836     m[2] = m0.z;
837     m[3] = m0.w;
838     m[4] = m1.x;
839     m[5] = m1.y;
840     m[6] = m1.z;
841     m[7] = m1.w;
842     m[8] = m2.x;
843     m[9] = m2.y;
844     m[10] = m2.z;
845     m[11] = m2.w;
846 }
847
848 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12])
849 {
850     if(optixGetTransformListSize() == 0)
851     {
852         m[0] = 1.0f;
853         m[1] = 0.0f;

```



```

854         m[2] = 0.0f;
855         m[3] = 0.0f;
856         m[4] = 0.0f;
857         m[5] = 1.0f;
858         m[6] = 0.0f;
859         m[7] = 0.0f;
860         m[8] = 0.0f;
861         m[9] = 0.0f;
862         m[10] = 1.0f;
863         m[11] = 0.0f;
864         return;
865     }
866
867     float4 m0, m1, m2;
868     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);
869     m[0] = m0.x;
870     m[1] = m0.y;
871     m[2] = m0.z;
872     m[3] = m0.w;
873     m[4] = m1.x;
874     m[5] = m1.y;
875     m[6] = m1.z;
876     m[7] = m1.w;
877     m[8] = m2.x;
878     m[9] = m2.y;
879     m[10] = m2.z;
880     m[11] = m2.w;
881 }
882
883 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point)
884 {
885     if(optixGetTransformListSize() == 0)
886         return point;
887
888     float4 m0, m1, m2;
889     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
890     return optix_impl::optixTransformPoint(m0, m1, m2, point);
891 }
892
893 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec)
894 {
895     if(optixGetTransformListSize() == 0)
896         return vec;
897
898     float4 m0, m1, m2;
899     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
900     return optix_impl::optixTransformVector(m0, m1, m2, vec);
901 }
902
903 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal)
904 {
905     if(optixGetTransformListSize() == 0)
906         return normal;
907
908     float4 m0, m1, m2;
909     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2); // inverse of
910     optixGetWorldToObjectTransformMatrix()
911     return optix_impl::optixTransformNormal(m0, m1, m2, normal);
912 }
913
914 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point)
915 {
916     if(optixGetTransformListSize() == 0)
917         return point;
918
919     float4 m0, m1, m2;
920     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);

```

```

920     return optix_impl::optixTransformPoint(m0, m1, m2, point);
921 }
922
923 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec)
924 {
925     if(optixGetTransformListSize() == 0)
926         return vec;
927
928     float4 m0, m1, m2;
929     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);
930     return optix_impl::optixTransformVector(m0, m1, m2, vec);
931 }
932
933 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal)
934 {
935     if(optixGetTransformListSize() == 0)
936         return normal;
937
938     float4 m0, m1, m2;
939     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2); // inverse of
optixGetObjectToWorldTransformMatrix()
940     return optix_impl::optixTransformNormal(m0, m1, m2, normal);
941 }
942
943 static __forceinline__ __device__ unsigned int optixGetTransformListSize()
944 {
945     unsigned int u0;
946     asm("call (%0), _optix_get_transform_list_size, ();" : "=r"(u0) :);
947     return u0;
948 }
949
950 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int index)
951 {
952     unsigned long long u0;
953     asm("call (%0), _optix_get_transform_list_handle, (%1);" : "=l"(u0) : "r"(index) :);
954     return u0;
955 }
956
957 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle)
958 {
959     int i0;
960     asm("call (%0), _optix_get_transform_type_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
961     return (OptixTransformType)i0;
962 }
963
964 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle)
965 {
966     unsigned long long ptr;
967     asm("call (%0), _optix_get_static_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
968     return (const OptixStaticTransform*)ptr;
969 }
970
971 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle)
972 {
973     unsigned long long ptr;
974     asm("call (%0), _optix_get_srt_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
975     return (const OptixSRTMotionTransform*)ptr;
976 }
977
978 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle)
979 {
980     unsigned long long ptr;
981     asm("call (%0), _optix_get_matrix_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);

```

```

982     return (const OptixMatrixMotionTransform*)ptr;
983 }
984
985 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle)
986 {
987     int i0;
988     asm("call (%0), _optix_get_instance_id_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
989     return i0;
990 }
991
992 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle(OptixTraversableHandle handle)
993 {
994     unsigned long long i0;
995     asm("call (%0), _optix_get_instance_child_from_handle, (%1);" : "=l"(i0) : "l"(handle) :);
996     return (OptixTraversableHandle)i0;
997 }
998
999 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle)
1000 {
1001     unsigned long long ptr;
1002     asm("call (%0), _optix_get_instance_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
1003     return (const float4*)ptr;
1004 }
1005
1006 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle)
1007 {
1008     unsigned long long ptr;
1009     asm("call (%0), _optix_get_instance_inverse_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle)
:);
1010     return (const float4*)ptr;
1011 }
1012
1013 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind)
1014 {
1015     int ret;
1016     asm volatile(
1017         "call (%0), _optix_report_intersection_0"
1018         ", (%1, %2);"
1019         : "=r"(ret)
1020         : "f"(hitT), "r"(hitKind)
1021         :);
1022     return ret;
1023 }
1024
1025 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0)
1026 {
1027     int ret;
1028     asm volatile(
1029         "call (%0), _optix_report_intersection_1"
1030         ", (%1, %2, %3);"
1031         : "=r"(ret)
1032         : "f"(hitT), "r"(hitKind), "r"(a0)
1033         :);
1034     return ret;
1035 }
1036
1037 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1)
1038 {
1039     int ret;
1040     asm volatile(
1041         "call (%0), _optix_report_intersection_2"

```

```

1042         ", (%1, %2, %3, %4);"
1043         : "=r"(ret)
1044         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1)
1045         :);
1046     return ret;
1047 }
1048
1049 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1, unsigned int a2)
1050 {
1051     int ret;
1052     asm volatile(
1053         "call (%0), _optix_report_intersection_3"
1054         ", (%1, %2, %3, %4, %5);"
1055         : "=r"(ret)
1056         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2)
1057         :);
1058     return ret;
1059 }
1060
1061 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
1062  unsigned int hitKind,
1063  unsigned int a0,
1064  unsigned int a1,
1065  unsigned int a2,
1066  unsigned int a3)
1067 {
1068     int ret;
1069     asm volatile(
1070         "call (%0), _optix_report_intersection_4"
1071         ", (%1, %2, %3, %4, %5, %6);"
1072         : "=r"(ret)
1073         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3)
1074         :);
1075     return ret;
1076 }
1077
1078 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
1079  unsigned int hitKind,
1080  unsigned int a0,
1081  unsigned int a1,
1082  unsigned int a2,
1083  unsigned int a3,
1084  unsigned int a4)
1085 {
1086     int ret;
1087     asm volatile(
1088         "call (%0), _optix_report_intersection_5"
1089         ", (%1, %2, %3, %4, %5, %6, %7);"
1090         : "=r"(ret)
1091         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4)
1092         :);
1093     return ret;
1094 }
1095
1096 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
1097  unsigned int hitKind,
1098  unsigned int a0,
1099  unsigned int a1,
1100  unsigned int a2,
1101  unsigned int a3,
1102  unsigned int a4,
1103  unsigned int a5)
1104 {
1105     int ret;
1106     asm volatile(
1107         "call (%0), _optix_report_intersection_6"

```

```

1108     ", (%1, %2, %3, %4, %5, %6, %7, %8);"
1109     : "=r"(ret)
1110     : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5)
1111     :);
1112     return ret;
1113 }
1114
1115 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
1116  unsigned int hitKind,
1117  unsigned int a0,
1118  unsigned int a1,
1119  unsigned int a2,
1120  unsigned int a3,
1121  unsigned int a4,
1122  unsigned int a5,
1123  unsigned int a6)
1124 {
1125     int ret;
1126     asm volatile(
1127         "call (%0), _optix_report_intersection_7"
1128         ", (%1, %2, %3, %4, %5, %6, %7, %8, %9);"
1129         : "=r"(ret)
1130         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6)
1131         :);
1132     return ret;
1133 }
1134
1135 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
1136  unsigned int hitKind,
1137  unsigned int a0,
1138  unsigned int a1,
1139  unsigned int a2,
1140  unsigned int a3,
1141  unsigned int a4,
1142  unsigned int a5,
1143  unsigned int a6,
1144  unsigned int a7)
1145 {
1146     int ret;
1147     asm volatile(
1148         "call (%0), _optix_report_intersection_8"
1149         ", (%1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
1150         : "=r"(ret)
1151         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6), "r"(a7)
1152         :);
1153     return ret;
1154 }
1155
1156 #define OPTIX_DEFINE_optixGetAttribute_BODY(which)
1157 \
1158 unsigned int ret;
1159 \
1160 asm("call (%0), _optix_get_attribute_" #which ", ();" : "=r"(ret) :);
1161 \
1162 return ret;
1163
1164 static __forceinline__ __device__ unsigned int optixGetAttribute_0()
1165 {
1166     OPTIX_DEFINE_optixGetAttribute_BODY(0);
1167 }
1168
1169 static __forceinline__ __device__ unsigned int optixGetAttribute_1()
1170 {
1171     OPTIX_DEFINE_optixGetAttribute_BODY(1);
1172 }
1173
1174 static __forceinline__ __device__ unsigned int optixGetAttribute_2()

```

```

1172 {
1173     OPTIX_DEFINE_optixGetAttribute_BODY(2);
1174 }
1175
1176 static __forceinline__ __device__ unsigned int optixGetAttribute_3()
1177 {
1178     OPTIX_DEFINE_optixGetAttribute_BODY(3);
1179 }
1180
1181 static __forceinline__ __device__ unsigned int optixGetAttribute_4()
1182 {
1183     OPTIX_DEFINE_optixGetAttribute_BODY(4);
1184 }
1185
1186 static __forceinline__ __device__ unsigned int optixGetAttribute_5()
1187 {
1188     OPTIX_DEFINE_optixGetAttribute_BODY(5);
1189 }
1190
1191 static __forceinline__ __device__ unsigned int optixGetAttribute_6()
1192 {
1193     OPTIX_DEFINE_optixGetAttribute_BODY(6);
1194 }
1195
1196 static __forceinline__ __device__ unsigned int optixGetAttribute_7()
1197 {
1198     OPTIX_DEFINE_optixGetAttribute_BODY(7);
1199 }
1200
1201 #undef OPTIX_DEFINE_optixGetAttribute_BODY
1202
1203 static __forceinline__ __device__ void optixTerminateRay()
1204 {
1205     asm volatile("call _optix_terminate_ray, ();");
1206 }
1207
1208 static __forceinline__ __device__ void optixIgnoreIntersection()
1209 {
1210     asm volatile("call _optix_ignore_intersection, ();");
1211 }
1212
1213 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex()
1214 {
1215     unsigned int u0;
1216     asm("call (%0), _optix_read_primitive_idx, ();" : "=r"(u0) :);
1217     return u0;
1218 }
1219
1220 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex()
1221 {
1222     unsigned int u0;
1223     asm("call (%0), _optix_read_sbt_gas_idx, ();" : "=r"(u0) :);
1224     return u0;
1225 }
1226
1227 static __forceinline__ __device__ unsigned int optixGetInstanceId()
1228 {
1229     unsigned int u0;
1230     asm("call (%0), _optix_read_instance_id, ();" : "=r"(u0) :);
1231     return u0;
1232 }
1233
1234 static __forceinline__ __device__ unsigned int optixGetInstanceIndex()
1235 {
1236     unsigned int u0;
1237     asm("call (%0), _optix_read_instance_idx, ();" : "=r"(u0) :);
1238     return u0;

```

```

1239 }
1240
1241 static __forceinline__ __device__ unsigned int optixGetHitKind()
1242 {
1243     unsigned int u0;
1244     asm("call (%0), _optix_get_hit_kind, ();" : "=r"(u0) :);
1245     return u0;
1246 }
1247
1248 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind)
1249 {
1250     unsigned int u0;
1251     asm("call (%0), _optix_get_primitive_type_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));
1252     return (OptixPrimitiveType)u0;
1253 }
1254
1255 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind)
1256 {
1257     unsigned int u0;
1258     asm("call (%0), _optix_get_backface_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));
1259     return (u0 == 0x1);
1260 }
1261
1262 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind)
1263 {
1264     return !optixIsBackFaceHit(hitKind);
1265 }
1266
1267
1268 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType()
1269 {
1270     return optixGetPrimitiveType(optixGetHitKind());
1271 }
1272
1273 static __forceinline__ __device__ bool optixIsBackFaceHit()
1274 {
1275     return optixIsBackFaceHit(optixGetHitKind());
1276 }
1277
1278 static __forceinline__ __device__ bool optixIsFrontFaceHit()
1279 {
1280     return optixIsFrontFaceHit(optixGetHitKind());
1281 }
1282
1283 static __forceinline__ __device__ bool optixIsTriangleHit()
1284 {
1285     return optixIsTriangleFrontFaceHit() || optixIsTriangleBackFaceHit();
1286 }
1287
1288 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit()
1289 {
1290     return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE;
1291 }
1292
1293 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit()
1294 {
1295     return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_BACK_FACE;
1296 }
1297
1298 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit()
1299 {
1300     return optixGetPrimitiveType(optixGetHitKind()) ==
1301     OPTIX_PRIMITIVE_TYPE_DISPLACED_MICROMESH_TRIANGLE;
1302 }
1303
1304 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleFrontFaceHit()
1305 {

```

```

1305     return optixIsDisplacedMicromeshTriangleHit() && optixIsFrontFaceHit();
1306 }
1307
1308 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleBackFaceHit()
1309 {
1310     return optixIsDisplacedMicromeshTriangleHit() && optixIsBackFaceHit();
1311 }
1312
1313 static __forceinline__ __device__ float optixGetCurveParameter()
1314 {
1315     float f0;
1316     asm("call (%0), _optix_get_curve_parameter, ();" : "=f"(f0) :);
1317     return f0;
1318 }
1319
1320 static __forceinline__ __device__ float2 optixGetRibbonParameters()
1321 {
1322     float f0, f1;
1323     asm("call (%0, %1), _optix_get_ribbon_parameters, ();" : "=f"(f0), "=f"(f1) :);
1324     return make_float2(f0, f1);
1325 }
1326
1327 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics()
1328 {
1329     float f0, f1;
1330     asm("call (%0, %1), _optix_get_triangle_barycentrics, ();" : "=f"(f0), "=f"(f1) :);
1331     return make_float2(f0, f1);
1332 }
1333
1334 static __forceinline__ __device__ uint3 optixGetLaunchIndex()
1335 {
1336     unsigned int u0, u1, u2;
1337     asm("call (%0), _optix_get_launch_index_x, ();" : "=r"(u0) :);
1338     asm("call (%0), _optix_get_launch_index_y, ();" : "=r"(u1) :);
1339     asm("call (%0), _optix_get_launch_index_z, ();" : "=r"(u2) :);
1340     return make_uint3(u0, u1, u2);
1341 }
1342
1343 static __forceinline__ __device__ uint3 optixGetLaunchDimensions()
1344 {
1345     unsigned int u0, u1, u2;
1346     asm("call (%0), _optix_get_launch_dimension_x, ();" : "=r"(u0) :);
1347     asm("call (%0), _optix_get_launch_dimension_y, ();" : "=r"(u1) :);
1348     asm("call (%0), _optix_get_launch_dimension_z, ();" : "=r"(u2) :);
1349     return make_uint3(u0, u1, u2);
1350 }
1351
1352 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer()
1353 {
1354     unsigned long long ptr;
1355     asm("call (%0), _optix_get_sbt_data_ptr_64, ();" : "=l"(ptr) :);
1356     return (CUdeviceptr)ptr;
1357 }
1358
1359 static __forceinline__ __device__ void optixThrowException(int exceptionCode)
1360 {
1361     asm volatile(
1362         "call _optix_throw_exception_0, (%0);"
1363         : /* no return value */
1364         : "r"(exceptionCode)
1365         :);
1366 }
1367
1368 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0)
1369 {
1370     asm volatile(

```



```

1371     "call _optix_throw_exception_1, (%0, %1);"
1372     : /* no return value */
1373     : "r"(exceptionCode), "r"(exceptionDetail0)
1374     :);
1375 }
1376
1377 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1)
1378 {
1379     asm volatile(
1380         "call _optix_throw_exception_2, (%0, %1, %2);"
1381         : /* no return value */
1382         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1)
1383         :);
1384 }
1385
1386 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
1387 {
1388     asm volatile(
1389         "call _optix_throw_exception_3, (%0, %1, %2, %3);"
1390         : /* no return value */
1391         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2)
1392         :);
1393 }
1394
1395 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3)
1396 {
1397     asm volatile(
1398         "call _optix_throw_exception_4, (%0, %1, %2, %3, %4);"
1399         : /* no return value */
1400         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3)
1401         :);
1402 }
1403
1404 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3, unsigned int exceptionDetail4)
1405 {
1406     asm volatile(
1407         "call _optix_throw_exception_5, (%0, %1, %2, %3, %4, %5);"
1408         : /* no return value */
1409         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4)
1410         :);
1411 }
1412
1413 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
1414 {
1415     asm volatile(
1416         "call _optix_throw_exception_6, (%0, %1, %2, %3, %4, %5, %6);"
1417         : /* no return value */
1418         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5)
1419         :);
1420 }
1421
1422 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int
exceptionDetail6)
1423 {

```

```

1424     asm volatile(
1425         "call _optix_throw_exception_7, (%0, %1, %2, %3, %4, %5, %6, %7);"
1426         : /* no return value */
1427         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
1428         : "r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6)
1429     );
1430 }
1431 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int
exceptionDetail6, unsigned int exceptionDetail7)
1432 {
1433     asm volatile(
1434         "call _optix_throw_exception_8, (%0, %1, %2, %3, %4, %5, %6, %7, %8);"
1435         : /* no return value */
1436         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
1437         : "r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6),
1438         : "r"(exceptionDetail7)
1439     );
1440 }
1441 static __forceinline__ __device__ int optixGetExceptionCode()
1442 {
1443     int s0;
1444     asm("call (%0), _optix_get_exception_code, ();" : "=r"(s0) :);
1445     return s0;
1446 }
1447 #define OPTIX_DEFINE_optixGetExceptionDetail_BODY(which)
1448 \
1449 unsigned int ret;
1450 \
1451 asm("call (%0), _optix_get_exception_detail_" #which ", ();" : "=r"(ret) :);
1452 \
1453 return ret;
1454 }
1455 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0()
1456 {
1457     OPTIX_DEFINE_optixGetExceptionDetail_BODY(0);
1458 }
1459 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1()
1460 {
1461     OPTIX_DEFINE_optixGetExceptionDetail_BODY(1);
1462 }
1463 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2()
1464 {
1465     OPTIX_DEFINE_optixGetExceptionDetail_BODY(2);
1466 }
1467 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3()
1468 {
1469     OPTIX_DEFINE_optixGetExceptionDetail_BODY(3);
1470 }
1471 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4()
1472 {
1473     OPTIX_DEFINE_optixGetExceptionDetail_BODY(4);
1474 }
1475 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5()
1476 {
1477     OPTIX_DEFINE_optixGetExceptionDetail_BODY(5);
1478 }
1479 }
1480 }
1481

```

```

1482 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6()
1483 {
1484     OPTIX_DEFINE_optixGetExceptionDetail_BODY(6);
1485 }
1486
1487 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7()
1488 {
1489     OPTIX_DEFINE_optixGetExceptionDetail_BODY(7);
1490 }
1491
1492 #undef OPTIX_DEFINE_optixGetExceptionDetail_BODY
1493
1494 static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable()
1495 {
1496     unsigned long long handle;
1497     asm("call (%0), _optix_get_exception_invalid_traversable, ();" : "=l"(handle) :);
1498     return (OptixTraversableHandle)handle;
1499 }
1500
1501 static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset()
1502 {
1503     int s0;
1504     asm("call (%0), _optix_get_exception_invalid_sbt_offset, ();" : "=r"(s0) :);
1505     return s0;
1506 }
1507
1508 static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay()
1509 {
1510     float rayOriginX, rayOriginY, rayOriginZ, rayDirectionX, rayDirectionY, rayDirectionZ, tmin, tmax,
    rayTime;
1511     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_exception_invalid_ray, ();"
1512         : "=f"(rayOriginX), "=f"(rayOriginY), "=f"(rayOriginZ), "=f"(rayDirectionX),
    "=f"(rayDirectionY),
1513         "=f"(rayDirectionZ), "=f"(tmin), "=f"(tmax), "=f"(rayTime)
1514         :);
1515     OptixInvalidRayExceptionDetails ray;
1516     ray.origin = make_float3(rayOriginX, rayOriginY, rayOriginZ);
1517     ray.direction = make_float3(rayDirectionX, rayDirectionY, rayDirectionZ);
1518     ray.tmin = tmin;
1519     ray.tmax = tmax;
1520     ray.time = rayTime;
1521     return ray;
1522 }
1523
1524 static __forceinline__ __device__ OptixParameterMismatchExceptionDetails
    optixGetExceptionParameterMismatch()
1525 {
1526     unsigned int expected, actual, sbtIdx;
1527     unsigned long long calleeName;
1528     asm(
1529         "call (%0, %1, %2, %3), _optix_get_exception_parameter_mismatch, ();"
1530         : "=r"(expected), "=r"(actual), "=r"(sbtIdx), "=l"(calleeName) :);
1531     OptixParameterMismatchExceptionDetails details;
1532     details.expectedParameterCount = expected;
1533     details.passedArgumentCount = actual;
1534     details.sbtIndex = sbtIdx;
1535     details.callableName = (char*)calleeName;
1536     return details;
1537 }
1538
1539 static __forceinline__ __device__ char* optixGetExceptionLineInfo()
1540 {
1541     unsigned long long ptr;
1542     asm("call (%0), _optix_get_exception_line_info, ();" : "=l"(ptr) :);
1543     return (char*)ptr;
1544 }
1545

```

```

1546 template <typename ReturnT, typename... ArgTypes>
1547 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args)
1548 {
1549     unsigned long long func;
1550     asm("call (%0), _optix_call_direct_callable, (%1);" : "=l"(func) : "r"(sbtIndex) :);
1551     using funcT = ReturnT (*)(ArgTypes...);
1552     funcT call = (funcT)(func);
1553     return call(args...);
1554 }
1555
1556 template <typename ReturnT, typename... ArgTypes>
1557 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes... args)
1558 {
1559     unsigned long long func;
1560     asm("call (%0), _optix_call_continuation_callable, (%1);" : "=l"(func) : "r"(sbtIndex) :);
1561     using funcT = ReturnT (*)(ArgTypes...);
1562     funcT call = (funcT)(func);
1563     return call(args...);
1564 }
1565
1566 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int
texInfo, float x, float y, unsigned int* singleMipLevel)
1567 {
1568     uint4 result;
1569     unsigned long long resultPtr = reinterpret_cast<unsigned long long>(&result);
1570     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
1571     // Cast float args to integers, because the intrinsics take .b32 arguments when compiled to PTX.
1572     asm volatile(
1573         "call _optix_tex_footprint_2d_v2"
1574         ", (%0, %1, %2, %3, %4, %5);"
1575         :
1576         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
1577         "l"(singleMipLevelPtr), "l"(resultPtr)
1578         :);
1579     return result;
1580 }
1581
1582 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
1583     unsigned int texInfo,
1584     float x,
1585     float y,
1586     float dPdx_x,
1587     float dPdx_y,
1588     float dPdy_x,
1589     float dPdy_y,
1590     bool coarse,
1591     unsigned int* singleMipLevel)
1592 {
1593     uint4 result;
1594     unsigned long long resultPtr = reinterpret_cast<unsigned long long>(&result);
1595     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
1596     // Cast float args to integers, because the intrinsics take .b32 arguments when compiled to PTX.
1597     asm volatile(
1598         "call _optix_tex_footprint_2d_grad_v2"
1599         ", (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
1600         :
1601         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
1602         "r"(__float_as_uint(dPdx_x)), "r"(__float_as_uint(dPdx_y)), "r"(__float_as_uint(dPdy_x)),
1603         "r"(__float_as_uint(dPdy_y)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
1604         "l"(resultPtr)
1605         :);
1606     return result;
1607 }
1608
1609 static __forceinline__ __device__ uint4
1610 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool

```

```

coarse, unsigned int* singleMipLevel)
1611 {
1612     uint4          result;
1613     unsigned long long resultPtr      = reinterpret_cast<unsigned long long>(&result);
1614     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
1615     // Cast float args to integers, because the intrinsics take .b32 arguments when compiled to PTX.
1616     asm volatile(
1617         "call _optix_tex_footprint_2d_lod_v2"
1618         ", (%0, %1, %2, %3, %4, %5, %6, %7);"
1619         :
1620         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
1621         "r"(__float_as_uint(level)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
1622         "l"(resultPtr)
1623         :);
1624     return result;
1625 }
1626 #endif // OPTIX_DEVICE_IMPL_H

```

## 8.3 optix\_device\_impl\_exception.h File Reference

### Namespaces

- namespace `optix_impl`

### Functions

- static `__forceinline__ __device__ void optix_impl::optixDumpStaticTransformFromHandle (OptixTraversableHandle handle)`
- static `__forceinline__ __device__ void optix_impl::optixDumpMotionMatrixTransformFromHandle (OptixTraversableHandle handle)`
- static `__forceinline__ __device__ void optix_impl::optixDumpSrtMatrixTransformFromHandle (OptixTraversableHandle handle)`
- static `__forceinline__ __device__ void optix_impl::optixDumpInstanceFromHandle (OptixTraversableHandle handle)`
- static `__forceinline__ __device__ void optix_impl::optixDumpTransform (OptixTraversableHandle handle)`
- static `__forceinline__ __device__ void optix_impl::optixDumpTransformList ()`
- static `__forceinline__ __device__ void optix_impl::optixDumpExceptionDetails ()`

### 8.3.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Device side implementation for exception helper function.

## 8.4 optix\_device\_impl\_exception.h

[Go to the documentation of this file.](#)

```

1 /*
2 * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
3 *
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly

```

```

8 * prohibited.
9 *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE. IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
21 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
22 #error("optix_device_impl_exception.h is an internal header file and must not be used directly. Please
23 use optix_device.h or optix.h instead.")
24 #endif
25
26 #ifndef OPTIX_DEVICE_IMPL_EXCEPTION_H
27 #define OPTIX_DEVICE_IMPL_EXCEPTION_H
28
29 #if !defined(__CUDACC_RTC__)
30 #include <stdio> /* for printf */
31 #endif
32
33 namespace optix_impl {
34
35     static __forceinline__ __device__ void optixDumpStaticTransformFromHandle(OptixTraversableHandle
36 handle)
37     {
38         const OptixStaticTransform* traversable = optixGetStaticTransformFromHandle(handle);
39         if(traversable)
40         {
41             const uint3 index = optixGetLaunchIndex();
42             printf("(%4i,%4i,%4i)    OptixStaticTransform%p = {\n"
43                 "                                child      = %p,\n"
44                 "                                transform   = { %f,%f,%f,%f,\n"
45                 "  %f,%f,%f,%f,\n"
46                 "  %f,%f,%f,%f } }\n",
47                 index.x, index.y, index.z,
48                 traversable,
49                 (void*)traversable->child,
50                 traversable->transform[0], traversable->transform[1], traversable->transform[2],
51                 traversable->transform[3],
52                 traversable->transform[4], traversable->transform[5], traversable->transform[6],
53                 traversable->transform[7],
54                 traversable->transform[8], traversable->transform[9], traversable->transform[10],
55                 traversable->transform[11]);
56         }
57     }
58
59     static __forceinline__ __device__ void
60 optixDumpMotionMatrixTransformFromHandle(OptixTraversableHandle handle)
61     {
62         const OptixMatrixMotionTransform* traversable = optixGetMatrixMotionTransformFromHandle(handle);
63         if(traversable)
64         {
65             const uint3 index = optixGetLaunchIndex();
66             printf("(%4i,%4i,%4i)    OptixMatrixMotionTransform%p = {\n"
67                 "                                child      = %p,\n"
68                 "                                motionOptions = { numKeys = %i, flags = %i, timeBegin = %f,
69 timeEnd = %f },\n"
70                 "                                transform   = { { %f,%f,%f,%f,\n"
71                 "  %f,%f,%f,%f,\n"
72                 "  %f,%f,%f,%f }, ... } }\n",
73                 index.x, index.y, index.z,
74                 traversable,
75                 (void*)traversable->child,
76                 traversable->motionOptions.numKeys, traversable->motionOptions.flags,
77                 traversable->motionOptions.timeBegin, traversable->motionOptions.timeEnd,
78                 traversable->transform[0][0], traversable->transform[0][1], traversable->transform[0][2],
79                 traversable->transform[0][3],
80                 traversable->transform[1][0], traversable->transform[1][1], traversable->transform[1][2],
81                 traversable->transform[1][3],
82                 traversable->transform[2][0], traversable->transform[2][1], traversable->transform[2][2],
83                 traversable->transform[2][3],
84                 traversable->transform[3][0], traversable->transform[3][1], traversable->transform[3][2],
85                 traversable->transform[3][3],
86                 traversable->transform[4][0], traversable->transform[4][1], traversable->transform[4][2],
87                 traversable->transform[4][3],
88                 traversable->transform[5][0], traversable->transform[5][1], traversable->transform[5][2],
89                 traversable->transform[5][3],
90                 traversable->transform[6][0], traversable->transform[6][1], traversable->transform[6][2],
91                 traversable->transform[6][3],
92                 traversable->transform[7][0], traversable->transform[7][1], traversable->transform[7][2],
93                 traversable->transform[7][3],
94                 traversable->transform[8][0], traversable->transform[8][1], traversable->transform[8][2],
95                 traversable->transform[8][3],
96                 traversable->transform[9][0], traversable->transform[9][1], traversable->transform[9][2],
97                 traversable->transform[9][3],
98                 traversable->transform[10][0], traversable->transform[10][1], traversable->transform[10][2],
99                 traversable->transform[10][3],
100                 traversable->transform[11][0], traversable->transform[11][1], traversable->transform[11][2],
101                 traversable->transform[11][3])

```

```

76         (void*)traversable->child,
77         (int)traversable->motionOptions.numKeys, (int)traversable->motionOptions.flags,
traversable->motionOptions.timeBegin, traversable->motionOptions.timeEnd,
78         traversable->transform[0][0], traversable->transform[0][1], traversable->transform[0][2],
traversable->transform[0][3],
79         traversable->transform[0][4], traversable->transform[0][5], traversable->transform[0][6],
traversable->transform[0][7],
80         traversable->transform[0][8], traversable->transform[0][9], traversable->transform[0][10],
traversable->transform[0][11]);
81     }
82 }
83
84 static __forceinline__ __device__ void optixDumpSrtMatrixTransformFromHandle(OptixTraversableHandle
handle)
85 {
86     const OptixSRTMotionTransform* traversable = optixGetSRTMotionTransformFromHandle(handle);
87     if(traversable)
88     {
89         const uint3 index = optixGetLaunchIndex();
90         printf("(%4i,%4i,%4i)    OptixSRTMotionTransform%p = {\n"
91             "                                child      = %p,\n"
92             "                                motionOptions = { numKeys = %i, flags = %i, timeBegin = %f,
timeEnd = %f },\n"
93             "                                srtData      = { { sx = %f,  a = %f,  b = %f, pvx = %f,\n"
94             "                                sy = %f,  c = %f, pvy = %f,  sz = %f,\n"
95             "                                pvz = %f, qx = %f,  qy = %f,  qz = %f,\n"
96             "                                qw = %f, tx = %f,  ty = %f,  tz = %f }, ... }\n",
97             index.x,index.y,index.z,
98             traversable,
99             (void*)traversable->child,
100             (int)traversable->motionOptions.numKeys, (int)traversable->motionOptions.flags,
traversable->motionOptions.timeBegin, traversable->motionOptions.timeEnd,
101             traversable->srtData[0].sx, traversable->srtData[0].a, traversable->srtData[0].b,
traversable->srtData[0].pvx,
102             traversable->srtData[0].sy, traversable->srtData[0].c,
traversable->srtData[0].pvy, traversable->srtData[0].sz,
103             traversable->srtData[0].pvz, traversable->srtData[0].qx, traversable->srtData[0].qy,
traversable->srtData[0].qz,
104             traversable->srtData[0].qw, traversable->srtData[0].tx, traversable->srtData[0].ty,
traversable->srtData[0].tz);
105     }
106 }
107
108 static __forceinline__ __device__ void optixDumpInstanceFromHandle(OptixTraversableHandle handle)
109 {
110     if(optixGetTransformTypeFromHandle(handle) == OPTIX_TRANSFORM_TYPE_INSTANCE)
111     {
112         unsigned int instanceId = optixGetInstanceIdFromHandle(handle);
113         const float4* transform = optixGetInstanceTransformFromHandle(handle);
114
115         const uint3 index = optixGetLaunchIndex();
116         printf("(%4i,%4i,%4i)    OptixInstance = {\n"
117             "                                instanceId = %i,\n"
118             "                                transform  = { %f,%f,%f,%f,\n"
119             "                                %f,%f,%f,%f,\n"
120             "                                %f,%f,%f,%f } }\n",
121             index.x,index.y,index.z,
122             instanceId,
123             transform[0].x, transform[0].y, transform[0].z,  transform[0].w,
124             transform[1].x, transform[1].y, transform[1].z,  transform[1].w,
125             transform[2].x, transform[2].y, transform[2].z,  transform[2].w);
126     }
127 }
128
129 static __forceinline__ __device__ void optixDumpTransform(OptixTraversableHandle handle)
130 {
131     const OptixTransformType type = optixGetTransformTypeFromHandle(handle);

```



```

132     const uint3 index = optixGetLaunchIndex();
133
134     switch(type)
135     {
136         case OPTIX_TRANSFORM_TYPE_NONE:
137             break;
138         case OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM:
139             optixDumpStaticTransformFromHandle(handle);
140             break;
141         case OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM:
142             optixDumpMotionMatrixTransformFromHandle(handle);
143             break;
144         case OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM:
145             optixDumpSrtMatrixTransformFromHandle(handle);
146             break;
147         case OPTIX_TRANSFORM_TYPE_INSTANCE:
148             optixDumpInstanceFromHandle(handle);
149             break;
150         default:
151             break;
152     }
153 }
154
155 static __forceinline__ __device__ void optixDumpTransformList()
156 {
157     const int tlistSize = optixGetTransformListSize();
158     const uint3 index = optixGetLaunchIndex();
159
160     printf("(%4i,%4i,%4i) transform list of size %i:\n", index.x,index.y,index.z, tlistSize);
161
162     for(unsigned int i = 0 ; i < tlistSize ; ++i)
163     {
164         OptixTraversableHandle handle = optixGetTransformListHandle(i);
165         printf("(%4i,%4i,%4i) transform[%i] = %p\n", index.x, index.y, index.z, i, (void*)handle);
166         optixDumpTransform(handle);
167     }
168 }
169
170 static __forceinline__ __device__ void optixDumpExceptionDetails()
171 {
172     bool dumpTlist = false;
173     const int exceptionCode = optixGetExceptionCode();
174     const uint3 index = optixGetLaunchIndex();
175
176     if(exceptionCode == OPTIX_EXCEPTION_CODE_STACK_OVERFLOW)
177     {
178         printf("(%4i,%4i,%4i) error: stack overflow\n", index.x,index.y,index.z);
179     }
180     else if(exceptionCode == OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED)
181     {
182         printf("(%4i,%4i,%4i) error: trace depth exceeded\n", index.x,index.y,index.z);
183     }
184     else if(exceptionCode == OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED)
185     {
186         printf("(%4i,%4i,%4i) error: traversal depth exceeded\n", index.x,index.y,index.z);
187         dumpTlist = true;
188     }
189     else if(exceptionCode == OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE)
190     {
191         OptixTraversableHandle handle = optixGetExceptionInvalidTraversable();
192         printf("(%4i,%4i,%4i) error: invalid traversable %p\n", index.x,index.y,index.z,
193 (void*)handle);
194         dumpTlist = true;
195     }
196     else if(exceptionCode == OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT)
197     {
198         int sbtOffset = optixGetExceptionInvalidSbtOffset();

```



```

198         printf("(%4i,%4i,%4i) error: invalid miss sbt of %i\n", index.x,index.y,index.z, sbtOffset);
199     }
200     else if(exceptionCode == OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT)
201     {
202         int sbtOffset = optixGetExceptionInvalidSbtOffset();
203         printf("(%4i,%4i,%4i) error: invalid hit sbt of %i at primitive with gas sbt index %i\n",
index.x,index.y,index.z, sbtOffset, optixGetSbtGASIndex());
204         dumpTlist = true;
205     }
206     else if(exceptionCode == OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE)
207     {
208         dumpTlist = true;
209         printf("(%4i,%4i,%4i) error: shader encountered unsupported builtin type\n"
                "                call location:  %s\n", index.x, index.y, index.z,
optixGetExceptionLineInfo());
210     }
211     }
212     else if(exceptionCode == OPTIX_EXCEPTION_CODE_INVALID_RAY)
213     {
214         OptixInvalidRayExceptionDetails ray = optixGetExceptionInvalidRay();
215         printf("(%4i,%4i,%4i) error: encountered an invalid ray:\n", index.x, index.y, index.z);
216         printf(
217             "        origin:          [%f, %f, %f]\n"
218             "        direction:         [%f, %f, %f]\n"
219             "        tmin:              %f\n"
220             "        tmax:              %f\n"
221             "        rayTime:           %f\n"
222             "        call location:      %s\n",
223             ray.origin.x, ray.origin.y, ray.origin.z, ray.direction.x, ray.direction.y,
224             ray.direction.z, ray.tmin, ray.tmax, ray.time, optixGetExceptionLineInfo());
225     }
226     else if(exceptionCode == OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH)
227     {
228         OptixParameterMismatchExceptionDetails details = optixGetExceptionParameterMismatch();
229         printf("(%4i,%4i,%4i) error: parameter mismatch in callable call.\n", index.x, index.y,
index.z);
230         printf(
231             "        passed packed arguments:      %u 32 Bit values\n"
232             "        expected packed parameters:    %u 32 Bit values\n"
233             "        SBT index:                    %u\n"
234             "        called function:                %s\n"
235             "        call location:                  %s\n",
236             details.passedArgumentCount, details.expectedParameterCount, details.sbtIndex,
237             details.callableName, optixGetExceptionLineInfo());
238     }
239     else if(exceptionCode == OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH)
240     {
241         dumpTlist = true;
242         printf("(%4i,%4i,%4i) error: mismatch between builtin IS shader and build input\n"
                "                call location:  %s\n", index.x,index.y,index.z, optixGetExceptionLineInfo());
243     }
244     }
245     else if(exceptionCode == OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT)
246     {
247         int sbtOffset = optixGetExceptionInvalidSbtOffset();
248         printf("(%4i,%4i,%4i) error: invalid sbt offset of %i for callable program\n", index.x,
index.y, index.z, sbtOffset);
249     }
250     else if(exceptionCode == OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD)
251     {
252         int sbtOffset = optixGetExceptionInvalidSbtOffset();
253         printf("(%4i,%4i,%4i) error: invalid sbt offset of %i for direct callable program\n",
index.x, index.y, index.z, sbtOffset);
254     }
255     else if(exceptionCode == OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD)
256     {
257         int sbtOffset = optixGetExceptionInvalidSbtOffset();
258         printf("(%4i,%4i,%4i) error: invalid sbt offset of %i for continuation callable program\n",
index.x, index.y, index.z, sbtOffset);

```

```

259     }
260     else if(exceptionCode == OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS)
261     {
262         OptixTraversableHandle handle = optixGetExceptionInvalidTraversable();
263         printf("(%4i,%4i,%4i) error: unsupported single GAS traversable graph %p\n",
index.x,index.y,index.z, (void*)handle);
264         dumpTlist = true;
265     }
266     else if((exceptionCode <= OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0) && (exceptionCode >=
OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2))
267     {
268         printf("(%4i,%4i,%4i) error: invalid value for argument %i\n", index.x,index.y,index.z,
-(exceptionCode - OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0));
269     }
270     else if(exceptionCode == OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS)
271     {
272         printf("(%4i,%4i,%4i) error: unsupported random data access\n", index.x,index.y,index.z);
273     }
274     else if(exceptionCode == OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH)
275     {
276         printf("(%4i,%4i,%4i) error: payload type mismatch between program and optixTrace call\n",
index.x,index.y,index.z);
277     }
278     else if(exceptionCode >= 0)
279     {
280         dumpTlist = true;
281         printf("(%4i,%4i,%4i) error: user exception with error code %i\n"
282             "      call location:  %s\n", index.x, index.y, index.z, exceptionCode,
optixGetExceptionLineInfo());
283     }
284     else
285     {
286         printf("(%4i,%4i,%4i) error: unknown exception with error code %i\n",
index.x,index.y,index.z, exceptionCode);
287     }
288
289     if(dumpTlist)
290         optixDumpTransformList();
291 }
292
293 } // namespace optix_impl
294
295 #endif // OPTIX_DEVICE_IMPL_EXCEPTION_H

```

## 8.5 optix\_device\_impl\_transformations.h File Reference

### Namespaces

- namespace `optix_impl`

### Functions

- static `__forceinline__ __device__ float4 optix_impl::optixAddFloat4` (const float4 &a, const float4 &b)
- static `__forceinline__ __device__ float4 optix_impl::optixMulFloat4` (const float4 &a, float b)
- static `__forceinline__ __device__ uint4 optix_impl::optixLdg` (unsigned long long addr)
- template<class T >  
static `__forceinline__ __device__ T optix_impl::optixLoadReadOnlyAlign16` (const T \*ptr)
- static `__forceinline__ __device__ float4 optix_impl::optixMultiplyRowMatrix` (const float4 vec, const float4 m0, const float4 m1, const float4 m2)
- static `__forceinline__ __device__ void optix_impl::optixGetMatrixFromSrt` (float4 &m0, float4 &m1, float4 &m2, const OptixSRTData &srt)

- static \_\_forceinline\_\_ \_\_device\_\_ void `optix_impl::optixInvertMatrix` (float4 &m0, float4 &m1, float4 &m2)
- static \_\_forceinline\_\_ \_\_device\_\_ void `optix_impl::optixLoadInterpolatedMatrixKey` (float4 &m0, float4 &m1, float4 &m2, const float4 \*matrix, const float t1)
- static \_\_forceinline\_\_ \_\_device\_\_ void `optix_impl::optixLoadInterpolatedSrtKey` (float4 &srt0, float4 &srt1, float4 &srt2, float4 &srt3, const float4 \*srt, const float t1)
- static \_\_forceinline\_\_ \_\_device\_\_ void `optix_impl::optixResolveMotionKey` (float &localt, int &key, const `OptixMotionOptions` &options, const float globalt)
- static \_\_forceinline\_\_ \_\_device\_\_ void `optix_impl::optixGetInterpolatedTransformation` (float4 &trf0, float4 &trf1, float4 &trf2, const `OptixMatrixMotionTransform` \*transformData, const float time)
- static \_\_forceinline\_\_ \_\_device\_\_ void `optix_impl::optixGetInterpolatedTransformation` (float4 &trf0, float4 &trf1, float4 &trf2, const `OptixSRTMotionTransform` \*transformData, const float time)
- static \_\_forceinline\_\_ \_\_device\_\_ void `optix_impl::optixGetInterpolatedTransformationFromHandle` (float4 &trf0, float4 &trf1, float4 &trf2, const `OptixTraversableHandle` handle, const float time, const bool objectToWorld)
- static \_\_forceinline\_\_ \_\_device\_\_ void `optix_impl::optixGetWorldToObjectTransformMatrix` (float4 &m0, float4 &m1, float4 &m2)
- static \_\_forceinline\_\_ \_\_device\_\_ void `optix_impl::optixGetObjectToWorldTransformMatrix` (float4 &m0, float4 &m1, float4 &m2)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 `optix_impl::optixTransformPoint` (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &p)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 `optix_impl::optixTransformVector` (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &v)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 `optix_impl::optixTransformNormal` (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &n)

### 8.5.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Device side implementation for transformation helper functions.

## 8.6 optix\_device\_impl\_transformations.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
3  *
4  * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5  * rights in and to this software, related documentation and any modifications thereto.
6  * Any use, reproduction, disclosure or distribution of this software and related
7  * documentation without an express license agreement from NVIDIA Corporation is strictly
8  * prohibited.
9  *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE. IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF

```

```

16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
21 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
22 #error("optix_device_impl_transformations.h is an internal header file and must not be used directly.
Please use optix_device.h or optix.h instead.")
23 #endif
24
25 #ifndef OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H
26 #define OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H
27
28 namespace optix_impl {
29
30 static __forceinline__ __device__ float4 optixAddFloat4(const float4& a, const float4& b)
31 {
32     return make_float4(a.x + b.x, a.y + b.y, a.z + b.z, a.w + b.w);
33 }
34
35 static __forceinline__ __device__ float4 optixMulFloat4(const float4& a, float b)
36 {
37     return make_float4(a.x * b, a.y * b, a.z * b, a.w * b);
38 }
39
40 static __forceinline__ __device__ uint4 optixLdg(unsigned long long addr)
41 {
42     const uint4* ptr;
43     asm volatile("cvta.to.global.u64 %0, %1;" : "=l"(ptr) : "l"(addr));
44     uint4 ret;
45     asm volatile("ld.global.v4.u32 {%0,%1,%2,%3}, [%4];"
46                 : "=r"(ret.x), "=r"(ret.y), "=r"(ret.z), "=r"(ret.w)
47                 : "l"(ptr));
48     return ret;
49 }
50
51 template <class T>
52 static __forceinline__ __device__ T optixLoadReadOnlyAlign16(const T* ptr)
53 {
54     T v;
55     for(int ofs = 0; ofs < sizeof(T); ofs += 16)
56         *(uint4*)((char*)&v + ofs) = optixLdg((unsigned long long)((char*)ptr + ofs));
57     return v;
58 }
59
60 // Multiplies the row vector vec with the 3x4 matrix with rows m0, m1, and m2
61 static __forceinline__ __device__ float4 optixMultiplyRowMatrix(const float4 vec, const float4 m0, const
float4 m1, const float4 m2)
62 {
63     float4 result;
64
65     result.x = vec.x * m0.x + vec.y * m1.x + vec.z * m2.x;
66     result.y = vec.x * m0.y + vec.y * m1.y + vec.z * m2.y;
67     result.z = vec.x * m0.z + vec.y * m1.z + vec.z * m2.z;
68     result.w = vec.x * m0.w + vec.y * m1.w + vec.z * m2.w + vec.w;
69
70     return result;
71 }
72
73 // Converts the SRT transformation srt into a 3x4 matrix with rows m0, m1, and m2
74 static __forceinline__ __device__ void optixGetMatrixFromSrt(float4& m0, float4& m1, float4& m2, const
OptixSRTData& srt)
75 {
76     const float4 q = {srt.qx, srt.qy, srt.qz, srt.qw};
77
78     // normalize
79     const float inv_sq1 = 1.f / (srt.qx * srt.qx + srt.qy * srt.qy + srt.qz * srt.qz + srt.qw * srt.qw);

```

```

88     const float4 nq      = optixMulFloat4(q, inv_sq1);
89
90     const float sqw = q.w * nq.w;
91     const float sqx = q.x * nq.x;
92     const float sqy = q.y * nq.y;
93     const float sqz = q.z * nq.z;
94
95     const float xy = q.x * nq.y;
96     const float zw = q.z * nq.w;
97     const float xz = q.x * nq.z;
98     const float yw = q.y * nq.w;
99     const float yz = q.y * nq.z;
100    const float xw = q.x * nq.w;
101
102    m0.x = (sqx - sqy - sqz + sqw);
103    m0.y = 2.0f * (xy - zw);
104    m0.z = 2.0f * (xz + yw);
105
106    m1.x = 2.0f * (xy + zw);
107    m1.y = (-sqx + sqy - sqz + sqw);
108    m1.z = 2.0f * (yz - xw);
109
110    m2.x = 2.0f * (xz - yw);
111    m2.y = 2.0f * (yz + xw);
112    m2.z = (-sqx - sqy + sqz + sqw);
113
114    m0.w = m0.x * srt.pvx + m0.y * srt.pvy + m0.z * srt.pvz + srt.tx;
115    m1.w = m1.x * srt.pvx + m1.y * srt.pvy + m1.z * srt.pvz + srt.ty;
116    m2.w = m2.x * srt.pvx + m2.y * srt.pvy + m2.z * srt.pvz + srt.tz;
117
118    m0.z = m0.x * srt.b + m0.y * srt.c + m0.z * srt.sz;
119    m1.z = m1.x * srt.b + m1.y * srt.c + m1.z * srt.sz;
120    m2.z = m2.x * srt.b + m2.y * srt.c + m2.z * srt.sz;
121
122    m0.y = m0.x * srt.a + m0.y * srt.sy;
123    m1.y = m1.x * srt.a + m1.y * srt.sy;
124    m2.y = m2.x * srt.a + m2.y * srt.sy;
125
126    m0.x = m0.x * srt.sx;
127    m1.x = m1.x * srt.sx;
128    m2.x = m2.x * srt.sx;
129 }
130
131 // Inverts a 3x4 matrix in place
132 static __forceinline__ __device__ void optixInvertMatrix(float4& m0, float4& m1, float4& m2)
133 {
134     const float det3 =
135         m0.x * (m1.y * m2.z - m1.z * m2.y) - m0.y * (m1.x * m2.z - m1.z * m2.x) + m0.z * (m1.x * m2.y -
136         m1.y * m2.x);
137     const float inv_det3 = 1.0f / det3;
138
139     float inv3[3][3];
140     inv3[0][0] = inv_det3 * (m1.y * m2.z - m2.y * m1.z);
141     inv3[0][1] = inv_det3 * (m0.z * m2.y - m2.z * m0.y);
142     inv3[0][2] = inv_det3 * (m0.y * m1.z - m1.y * m0.z);
143
144     inv3[1][0] = inv_det3 * (m1.z * m2.x - m2.z * m1.x);
145     inv3[1][1] = inv_det3 * (m0.x * m2.z - m2.x * m0.z);
146     inv3[1][2] = inv_det3 * (m0.z * m1.x - m1.z * m0.x);
147
148     inv3[2][0] = inv_det3 * (m1.x * m2.y - m2.x * m1.y);
149     inv3[2][1] = inv_det3 * (m0.y * m2.x - m2.y * m0.x);
150     inv3[2][2] = inv_det3 * (m0.x * m1.y - m1.x * m0.y);
151
152     const float b[3] = {m0.w, m1.w, m2.w};
153

```

```

154     m0.x = inv3[0][0];
155     m0.y = inv3[0][1];
156     m0.z = inv3[0][2];
157     m0.w = -inv3[0][0] * b[0] - inv3[0][1] * b[1] - inv3[0][2] * b[2];
158
159     m1.x = inv3[1][0];
160     m1.y = inv3[1][1];
161     m1.z = inv3[1][2];
162     m1.w = -inv3[1][0] * b[0] - inv3[1][1] * b[1] - inv3[1][2] * b[2];
163
164     m2.x = inv3[2][0];
165     m2.y = inv3[2][1];
166     m2.z = inv3[2][2];
167     m2.w = -inv3[2][0] * b[0] - inv3[2][1] * b[1] - inv3[2][2] * b[2];
168 }
169
170 static __forceinline__ __device__ void optixLoadInterpolatedMatrixKey(float4& m0, float4& m1, float4&
m2, const float4* matrix, const float t1)
171 {
172     m0 = optixLoadReadOnlyAlign16(&matrix[0]);
173     m1 = optixLoadReadOnlyAlign16(&matrix[1]);
174     m2 = optixLoadReadOnlyAlign16(&matrix[2]);
175
176     // The conditional prevents concurrent loads leading to spills
177     if(t1 > 0.0f)
178     {
179         const float t0 = 1.0f - t1;
180         m0 = optixAddFloat4(optixMulFloat4(m0, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[3]),
t1));
181         m1 = optixAddFloat4(optixMulFloat4(m1, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[4]),
t1));
182         m2 = optixAddFloat4(optixMulFloat4(m2, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[5]),
t1));
183     }
184 }
185
186 static __forceinline__ __device__ void optixLoadInterpolatedSrtKey(float4&      srt0,
187  float4&      srt1,
188  float4&      srt2,
189  float4&      srt3,
190  const float4* srt,
191  const float  t1)
192 {
193     srt0 = optixLoadReadOnlyAlign16(&srt[0]);
194     srt1 = optixLoadReadOnlyAlign16(&srt[1]);
195     srt2 = optixLoadReadOnlyAlign16(&srt[2]);
196     srt3 = optixLoadReadOnlyAlign16(&srt[3]);
197
198     // The conditional prevents concurrent loads leading to spills
199     if(t1 > 0.0f)
200     {
201         const float t0 = 1.0f - t1;
202         srt0 = optixAddFloat4(optixMulFloat4(srt0, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[4]),
t1));
203         srt1 = optixAddFloat4(optixMulFloat4(srt1, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[5]),
t1));
204         srt2 = optixAddFloat4(optixMulFloat4(srt2, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[6]),
t1));
205         srt3 = optixAddFloat4(optixMulFloat4(srt3, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[7]),
t1));
206
207         float inv_length = 1.f / sqrt(srt2.y * srt2.y + srt2.z * srt2.z + srt2.w * srt2.w + srt3.x *
srt3.x);
208         srt2.y *= inv_length;
209         srt2.z *= inv_length;
210         srt2.w *= inv_length;
211         srt3.x *= inv_length;

```

```

212     }
213 }
214
215 static __forceinline__ __device__ void optixResolveMotionKey(float& localt, int& key, const
OptixMotionOptions& options, const float globalt)
216 {
217     const float timeBegin    = options.timeBegin;
218     const float timeEnd      = options.timeEnd;
219     const float numIntervals = (float)(options.numKeys - 1);
220
221     // No need to check the motion flags. If data originates from a valid transform list handle, then
globalt is in
222     // range, or vanish flags are not set.
223
224     const float time = max(0.f, min(numIntervals, (globalt - timeBegin) * numIntervals / (timeEnd -
timeBegin)));
225     const float fltKey = floorf(time);
226
227     localt = time - fltKey;
228     key    = (int)fltKey;
229 }
230
231 // Returns the interpolated transformation matrix for a particular matrix motion transformation and point
in time.
232 static __forceinline__ __device__ void optixGetInterpolatedTransformation(float4&
trf0,
233                                     float4&                                trf1,
234                                     float4&                                trf2,
235                                     const OptixMatrixMotionTransform*
transformData,
236                                     const float                                time)
237 {
238     // Compute key and intra key time
239     float keyTime;
240     int key;
241     optixResolveMotionKey(keyTime, key, optixLoadReadOnlyAlign16(transformData).motionOptions, time);
242
243     // Get pointer to left key
244     const float4* transform = (const float4*)&transformData->transform[key][0];
245
246     // Load and interpolate matrix keys
247     optixLoadInterpolatedMatrixKey(trf0, trf1, trf2, transform, keyTime);
248 }
249
250 // Returns the interpolated transformation matrix for a particular SRT motion transformation and point in
time.
251 static __forceinline__ __device__ void optixGetInterpolatedTransformation(float4&
trf0,
252                                     float4&                                trf1,
253                                     float4&                                trf2,
254                                     const OptixSRTMotionTransform*
transformData,
255                                     const float                                time)
256 {
257     // Compute key and intra key time
258     float keyTime;
259     int key;
260     optixResolveMotionKey(keyTime, key, optixLoadReadOnlyAlign16(transformData).motionOptions, time);
261
262     // Get pointer to left key
263     const float4* dataPtr = reinterpret_cast<const float4*>(&transformData->srtData[key]);
264
265     // Load and interpolated SRT keys
266     float4 data[4];
267     optixLoadInterpolatedSrtKey(data[0], data[1], data[2], data[3], dataPtr, keyTime);
268
269     OptixSRTData srt = {data[0].x, data[0].y, data[0].z, data[0].w, data[1].x, data[1].y, data[1].z,

```

```

data[1].w,
270             data[2].x, data[2].y, data[2].z, data[2].w, data[3].x, data[3].y, data[3].z,
data[3].w};
271
272     // Convert SRT into a matrix
273     optixGetMatrixFromSrt(trf0, trf1, trf2, srt);
274 }
275
276 // Returns the interpolated transformation matrix for a particular traversable handle and point in time.
277 static __forceinline__ __device__ void optixGetInterpolatedTransformationFromHandle(float4&
trf0,
278                                     float4&
trf1,
279                                     float4&
trf2,
280                                     const
OptixTraversableHandle handle,
281                                     const float
time,
282                                     const bool objectToWorld)
283 {
284     const OptixTransformType type = optixGetTransformTypeFromHandle(handle);
285
286     if(type == OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM || type ==
OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM)
287     {
288         if(type == OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM)
289         {
290             const OptixMatrixMotionTransform* transformData =
optixGetMatrixMotionTransformFromHandle(handle);
291             optixGetInterpolatedTransformation(trf0, trf1, trf2, transformData, time);
292         }
293         else
294         {
295             const OptixSRTMotionTransform* transformData = optixGetSRTMotionTransformFromHandle(handle);
296             optixGetInterpolatedTransformation(trf0, trf1, trf2, transformData, time);
297         }
298
299         if(!objectToWorld)
300             optixInvertMatrix(trf0, trf1, trf2);
301     }
302     else if(type == OPTIX_TRANSFORM_TYPE_INSTANCE || type == OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM)
303     {
304         const float4* transform;
305
306         if(type == OPTIX_TRANSFORM_TYPE_INSTANCE)
307         {
308             transform = (objectToWorld) ? optixGetInstanceTransformFromHandle(handle) :
optixGetInstanceInverseTransformFromHandle(handle);
309         }
310         else
311         {
312             const OptixStaticTransform* traversable = optixGetStaticTransformFromHandle(handle);
313             transform = (const float4*)((objectToWorld) ? traversable->transform :
traversable->invTransform);
314         }
315
316         trf0 = optixLoadReadOnlyAlign16(&transform[0]);
317         trf1 = optixLoadReadOnlyAlign16(&transform[1]);
318         trf2 = optixLoadReadOnlyAlign16(&transform[2]);
319     }
320 }
321 else
322 {
323     trf0 = {1.0f, 0.0f, 0.0f, 0.0f};
324     trf1 = {0.0f, 1.0f, 0.0f, 0.0f};
325     trf2 = {0.0f, 0.0f, 1.0f, 0.0f};
326 }

```



```

327 }
328
329 // Returns the world-to-object transformation matrix resulting from the current transform stack and
current ray time.
330 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float4& m0, float4& m1,
float4& m2)
331 {
332     const unsigned int size = optixGetTransformListSize();
333     const float time = optixGetRayTime();
334
335     #pragma unroll 1
336     for(unsigned int i = 0; i < size; ++i)
337     {
338         OptixTraversableHandle handle = optixGetTransformListHandle(i);
339
340         float4 trf0, trf1, trf2;
341         optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
false);
342
343         if(i == 0)
344         {
345             m0 = trf0;
346             m1 = trf1;
347             m2 = trf2;
348         }
349         else
350         {
351             // m := trf * m
352             float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
353             m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
354             m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
355             m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
356         }
357     }
358 }
359
360 // Returns the object-to-world transformation matrix resulting from the current transform stack and
current ray time.
361 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float4& m0, float4& m1,
float4& m2)
362 {
363     const int size = optixGetTransformListSize();
364     const float time = optixGetRayTime();
365
366     #pragma unroll 1
367     for(int i = size - 1; i >= 0; --i)
368     {
369         OptixTraversableHandle handle = optixGetTransformListHandle(i);
370
371         float4 trf0, trf1, trf2;
372         optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
true);
373
374         if(i == size - 1)
375         {
376             m0 = trf0;
377             m1 = trf1;
378             m2 = trf2;
379         }
380         else
381         {
382             // m := trf * m
383             float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
384             m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
385             m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
386             m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
387         }

```

```

388     }
389 }
390
391 // Multiplies the 3x4 matrix with rows m0, m1, m2 with the point p.
392 static __forceinline__ __device__ float3 optixTransformPoint(const float4& m0, const float4& m1, const
float4& m2, const float3& p)
393 {
394     float3 result;
395     result.x = m0.x * p.x + m0.y * p.y + m0.z * p.z + m0.w;
396     result.y = m1.x * p.x + m1.y * p.y + m1.z * p.z + m1.w;
397     result.z = m2.x * p.x + m2.y * p.y + m2.z * p.z + m2.w;
398     return result;
399 }
400
401 // Multiplies the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the vector v.
402 static __forceinline__ __device__ float3 optixTransformVector(const float4& m0, const float4& m1, const
float4& m2, const float3& v)
403 {
404     float3 result;
405     result.x = m0.x * v.x + m0.y * v.y + m0.z * v.z;
406     result.y = m1.x * v.x + m1.y * v.y + m1.z * v.z;
407     result.z = m2.x * v.x + m2.y * v.y + m2.z * v.z;
408     return result;
409 }
410
411 // Multiplies the transpose of the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the
normal n.
412 // Note that the given matrix is supposed to be the inverse of the actual transformation matrix.
413 static __forceinline__ __device__ float3 optixTransformNormal(const float4& m0, const float4& m1, const
float4& m2, const float3& n)
414 {
415     float3 result;
416     result.x = m0.x * n.x + m1.x * n.y + m2.x * n.z;
417     result.y = m0.y * n.x + m1.y * n.y + m2.y * n.z;
418     result.z = m0.z * n.x + m1.z * n.y + m2.z * n.z;
419     return result;
420 }
421
422 } // namespace optix_impl
423
424 #endif // OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H

```

## 8.7 optix.h File Reference

### Macros

- #define `OPTIX_VERSION` 70700

### 8.7.1 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`

### 8.7.2 Macro Definition Documentation

#### 8.7.2.1 `OPTIX_VERSION`

```
#define OPTIX_VERSION 70700
```

The OptiX version.

- major = OPTIX\_VERSION/10000
- minor = (OPTIX\_VERSION%10000)/100
- micro = OPTIX\_VERSION%100

## 8.8 optix.h

[Go to the documentation of this file.](#)

```

1
2 /*
3 * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
4 *
5 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
6 * rights in and to this software, related documentation and any modifications thereto.
7 * Any use, reproduction, disclosure or distribution of this software and related
8 * documentation without an express license agreement from NVIDIA Corporation is strictly
9 * prohibited.
10 *
11 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
12 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
13 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
14 * PARTICULAR PURPOSE. IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
15 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
16 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
17 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
18 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
19 * SUCH DAMAGES
20 */
21
28
29 #ifndef OPTIX_OPTIX_H
30 #define OPTIX_OPTIX_H
31
37 #define OPTIX_VERSION 70700
38
39
40 #ifdef __CUDACC__
41 #include "optix_device.h"
42 #else
43 #include "optix_host.h"
44 #endif
45
46
47 #endif // OPTIX_OPTIX_H

```

## 8.9 optix\_denoiser\_tiling.h File Reference

### Classes

- struct [OptixUtilDenoiserImageTile](#)

### Functions

- [OptixResult optixUtilGetPixelStride](#) (const [OptixImage2D](#) &image, unsigned int &pixelStrideInBytes)
- [OptixResult optixUtilDenoiserSplitImage](#) (const [OptixImage2D](#) &input, const [OptixImage2D](#) &output, unsigned int overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight, std::vector< [OptixUtilDenoiserImageTile](#) > &tiles)
- [OptixResult optixUtilDenoiserInvokeTiled](#) ([OptixDenoiser](#) denoiser, CUstream stream, const [OptixDenoiserParams](#) \*params, CUdeviceptr denoiserState, size\_t denoiserStateSizeInBytes, const [OptixDenoiserGuideLayer](#) \*guideLayer, const [OptixDenoiserLayer](#) \*layers, unsigned int

numLayers, CUdeviceptr scratch, size\_t scratchSizeInBytes, unsigned int  
overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight)

### 8.9.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

## 8.10 optix\_denoiser\_tiling.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
3  *
4  * Redistribution and use in source and binary forms, with or without
5  * modification, are permitted provided that the following conditions
6  * are met:
7  * * Redistributions of source code must retain the above copyright
8  *   notice, this list of conditions and the following disclaimer.
9  * * Redistributions in binary form must reproduce the above copyright
10 *   notice, this list of conditions and the following disclaimer in the
11 *   documentation and/or other materials provided with the distribution.
12 * * Neither the name of NVIDIA CORPORATION nor the names of its
13 *   contributors may be used to endorse or promote products derived
14 *   from this software without specific prior written permission.
15 *
16 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS "AS IS" AND ANY
17 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
18 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
19 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
20 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
21 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
22 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
23 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
24 * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
25 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
26 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
27 */
28
29
30
31
32
33 #ifndef OPTIX_DENOISER_TILING_H
34 #define OPTIX_DENOISER_TILING_H
35
36 #include <optix.h>
37
38 #include <algorithm>
39 #include <vector>
40
41 #ifdef __cplusplus
42 extern "C" {
43 #endif
44
45 struct OptixUtilDenoiserImageTile
46 {
47     // input tile image
48     OptixImage2D input;
49
50     // output tile image
51     OptixImage2D output;
52
53     // overlap offsets, parameters for #optixUtilDenoiserInvoke

```

```

62     unsigned int inputOffsetX;
63     unsigned int inputOffsetY;
64 };
65
66 inline OptixResult optixUtilGetPixelStride(const OptixImage2D& image, unsigned int& pixelStrideInBytes)
67 {
68     pixelStrideInBytes = image.pixelStrideInBytes;
69     if(pixelStrideInBytes == 0)
70     {
71         switch(image.format)
72         {
73             case OPTIX_PIXEL_FORMAT_HALF1:
74                 pixelStrideInBytes = 1 * sizeof(short);
75                 break;
76             case OPTIX_PIXEL_FORMAT_HALF2:
77                 pixelStrideInBytes = 2 * sizeof(short);
78                 break;
79             case OPTIX_PIXEL_FORMAT_HALF3:
80                 pixelStrideInBytes = 3 * sizeof(short);
81                 break;
82             case OPTIX_PIXEL_FORMAT_HALF4:
83                 pixelStrideInBytes = 4 * sizeof(short);
84                 break;
85             case OPTIX_PIXEL_FORMAT_FLOAT1:
86                 pixelStrideInBytes = 1 * sizeof(float);
87                 break;
88             case OPTIX_PIXEL_FORMAT_FLOAT2:
89                 pixelStrideInBytes = 2 * sizeof(float);
90                 break;
91             case OPTIX_PIXEL_FORMAT_FLOAT3:
92                 pixelStrideInBytes = 3 * sizeof(float);
93                 break;
94             case OPTIX_PIXEL_FORMAT_FLOAT4:
95                 pixelStrideInBytes = 4 * sizeof(float);
96                 break;
97             case OPTIX_PIXEL_FORMAT_UCHAR3:
98                 pixelStrideInBytes = 3 * sizeof(char);
99                 break;
100             case OPTIX_PIXEL_FORMAT_UCHAR4:
101                 pixelStrideInBytes = 4 * sizeof(char);
102                 break;
103             case OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER:
104                 return OPTIX_ERROR_INVALID_VALUE;
105                 break;
106         }
107     }
108     return OPTIX_SUCCESS;
109 }
110
111 inline OptixResult optixUtilDenoiserSplitImage(
112     const OptixImage2D& input,
113     const OptixImage2D& output,
114     unsigned int overlapWindowSizeInPixels,
115     unsigned int tileWidth,
116     unsigned int tileHeight,
117     std::vector<OptixUtilDenoiserImageTile>& tiles)
118 {
119     if(tileWidth == 0 || tileHeight == 0)
120         return OPTIX_ERROR_INVALID_VALUE;
121
122     unsigned int inPixelStride, outPixelStride;
123     if(const OptixResult res = optixUtilGetPixelStride(input, inPixelStride))
124         return res;
125     if(const OptixResult res = optixUtilGetPixelStride(output, outPixelStride))
126         return res;
127
128     int inp_w = std::min(tileWidth + 2 * overlapWindowSizeInPixels, input.width);

```

```

145     int inp_h = std::min(tileHeight + 2 * overlapWindowSizeInPixels, input.height);
146     int inp_y = 0, copied_y = 0;
147
148     int upscaleX = output.width / input.width;
149     int upscaleY = output.height / input.height;
150
151     do
152     {
153         int inputOffsetY = inp_y == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_h -
154 ((int)input.height - inp_y));
155         int copy_y = inp_y == 0 ? std::min(input.height, tileHeight + overlapWindowSizeInPixels) :
156             std::min(tileHeight, input.height - copied_y);
157
158         int inp_x = 0, copied_x = 0;
159         do
160         {
161             int inputOffsetX = inp_x == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_w -
162 ((int)input.width - inp_x));
163             int copy_x = inp_x == 0 ? std::min(input.width, tileWidth + overlapWindowSizeInPixels) :
164                 std::min(tileWidth, input.width - copied_x);
165
166             OptixUtilDenoiserImageFile tile;
167             tile.input.data = input.data + (size_t)(inp_y - inputOffsetY) *
168 input.rowStrideInBytes
169             + (size_t)(inp_x - inputOffsetX) * inPixelStride;
170             tile.input.width = inp_w;
171             tile.input.height = inp_h;
172             tile.input.rowStrideInBytes = input.rowStrideInBytes;
173             tile.input.pixelStrideInBytes = input.pixelStrideInBytes;
174             tile.input.format = input.format;
175
176             tile.output.data = output.data + (size_t)(upscaleY * inp_y) *
177 output.rowStrideInBytes
178             + (size_t)(upscaleX * inp_x) * outPixelStride;
179             tile.output.width = upscaleX * copy_x;
180             tile.output.height = upscaleY * copy_y;
181             tile.output.rowStrideInBytes = output.rowStrideInBytes;
182             tile.output.pixelStrideInBytes = output.pixelStrideInBytes;
183             tile.output.format = output.format;
184
185             tile.inputOffsetX = inputOffsetX;
186             tile.inputOffsetY = inputOffsetY;
187
188             tiles.push_back(tile);
189
190             inp_x += inp_x == 0 ? tileWidth + overlapWindowSizeInPixels : tileWidth;
191             copied_x += copy_x;
192         } while(inp_x < static_cast<int>(input.width));
193
194         inp_y += inp_y == 0 ? tileHeight + overlapWindowSizeInPixels : tileHeight;
195         copied_y += copy_y;
196     } while(inp_y < static_cast<int>(input.height));
197
198     return OPTIX_SUCCESS;
199 }
200
201
202
203 inline OptixResult optixUtilDenoiserInvokeTiled(
204     OptixDenoiser          denoiser,
205     CUstream               stream,
206     const OptixDenoiserParams* params,
207     CUdeviceptr            denoiserState,
208     size_t                 denoiserStateSizeInBytes,
209     const OptixDenoiserGuideLayer* guideLayer,
210     const OptixDenoiserLayer* layers,
211     unsigned int            numLayers,

```

```

232         CUdeviceptr          scratch,
233         size_t               scratchSizeInBytes,
234         unsigned int         overlapWindowSizeInPixels,
235         unsigned int         tileWidth,
236         unsigned int         tileHeight)
237 {
238     if(!guideLayer || !layers)
239         return OPTIX_ERROR_INVALID_VALUE;
240
241     const unsigned int upscale = numLayers > 0 && layers[0].previousOutput.width == 2 *
layers[0].input.width ? 2 : 1;
242
243     std::vector<std::vector<OptixUtilDenoiserImageTile> tiles(numLayers);
244     std::vector<std::vector<OptixUtilDenoiserImageTile> prevTiles(numLayers);
245     for(unsigned int l = 0; l < numLayers; l++)
246     {
247         if(const OptixResult res = optixUtilDenoiserSplitImage(layers[l].input, layers[l].output,
248             overlapWindowSizeInPixels,
249             tileWidth, tileHeight, tiles[l]))
250             return res;
251
252         if(layers[l].previousOutput.data)
253         {
254             OptixImage2D dummyOutput = layers[l].previousOutput;
255             if(const OptixResult res = optixUtilDenoiserSplitImage(layers[l].previousOutput, dummyOutput,
256                 upscale * overlapWindowSizeInPixels,
257                 upscale * tileWidth, upscale * tileHeight,
prevTiles[l]))
258                 return res;
259         }
260     }
261
262     std::vector<OptixUtilDenoiserImageTile> albedoTiles;
263     if(guideLayer->albedo.data)
264     {
265         OptixImage2D dummyOutput = guideLayer->albedo;
266         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->albedo, dummyOutput,
267             overlapWindowSizeInPixels,
268             tileWidth, tileHeight, albedoTiles))
269             return res;
270     }
271
272     std::vector<OptixUtilDenoiserImageTile> normalTiles;
273     if(guideLayer->normal.data)
274     {
275         OptixImage2D dummyOutput = guideLayer->normal;
276         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->normal, dummyOutput,
277             overlapWindowSizeInPixels,
278             tileWidth, tileHeight, normalTiles))
279             return res;
280     }
281
282     std::vector<OptixUtilDenoiserImageTile> flowTiles;
283     if(guideLayer->flow.data)
284     {
285         OptixImage2D dummyOutput = guideLayer->flow;
286         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->flow, dummyOutput,
287             overlapWindowSizeInPixels,
288             tileWidth, tileHeight, flowTiles))
289             return res;
290     }
291
292     std::vector<OptixUtilDenoiserImageTile> flowTrustTiles;
293     if(guideLayer->flowTrustworthiness.data)
294     {
295         OptixImage2D dummyOutput = guideLayer->flowTrustworthiness;
296         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->flowTrustworthiness,

```

```

dummyOutput,
297                                     overlapWindowSizeInPixels,
298                                     tileWidth, tileHeight, flowTrustTiles))
299         return res;
300     }
301
302     std::vector<OptixUtilDenoiserImageTile> internalGuideLayerTiles;
303     if(guideLayer->previousOutputInternalGuideLayer.data && guideLayer->outputInternalGuideLayer.data)
304     {
305         if(const OptixResult res =
306         optixUtilDenoiserSplitImage(guideLayer->previousOutputInternalGuideLayer,
307                                     guideLayer->outputInternalGuideLayer,
308                                     upscale * overlapWindowSizeInPixels,
309                                     upscale * tileWidth, upscale * tileHeight,
310         internalGuideLayerTiles))
311             return res;
312     }
313
314     for(size_t t = 0; t < tiles[0].size(); t++)
315     {
316         std::vector<OptixDenoiserLayer> tlayers;
317         for(unsigned int l = 0; l < numLayers; l++)
318         {
319             OptixDenoiserLayer layer = {};
320             layer.input = (tiles[l])[t].input;
321             layer.output = (tiles[l])[t].output;
322             if(layers[l].previousOutput.data)
323                 layer.previousOutput = (prevTiles[l])[t].input;
324             layer.type = layers[l].type;
325             tlayers.push_back(layer);
326         }
327
328         OptixDenoiserGuideLayer gl = {};
329         if(guideLayer->albedo.data)
330             gl.albedo = albedoTiles[t].input;
331
332         if(guideLayer->normal.data)
333             gl.normal = normalTiles[t].input;
334
335         if(guideLayer->flow.data)
336             gl.flow = flowTiles[t].input;
337
338         if(guideLayer->flowTrustworthiness.data)
339             gl.flowTrustworthiness = flowTrustTiles[t].input;
340
341         if(guideLayer->previousOutputInternalGuideLayer.data)
342             gl.previousOutputInternalGuideLayer = internalGuideLayerTiles[t].input;
343
344         if(guideLayer->outputInternalGuideLayer.data)
345             gl.outputInternalGuideLayer = internalGuideLayerTiles[t].output;
346
347         if(const OptixResult res =
348         optixDenoiserInvoke(denoiser, stream, params, denoiserState, denoiserStateSizeInBytes,
349                             &gl, &tlayers[0], numLayers,
350                             (tiles[0])[t].inputOffsetX, (tiles[0])[t].inputOffsetY,
351                             scratch, scratchSizeInBytes))
352             return res;
353     }
354     return OPTIX_SUCCESS;
355 }
356 // end group optix_utilities
357 #ifdef __cplusplus
358 }
359 #endif
360
361 #endif // OPTIX_DENOISER_TILING_H

```



## 8.11 optix\_device.h File Reference

### Macros

- `#define __OPTIX_INCLUDE_INTERNAL_HEADERS__`

### Functions

- `template<typename... Payload>`  
`static __forceinline__ __device__ void optixTrace (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBTOffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)`
- `template<typename... Payload>`  
`static __forceinline__ __device__ void optixTrace (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBTOffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)`
- `static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_5 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_6 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_7 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_8 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_9 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_10 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_11 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_12 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_13 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_14 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_15 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_16 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_17 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_18 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_19 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_20 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_21 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_22 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_23 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_24 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_25 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_26 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_27 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_28 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_29 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_30 (unsigned int p)`
- `static __forceinline__ __device__ void optixSetPayload_31 (unsigned int p)`
- `static __forceinline__ __device__ unsigned int optixGetPayload_0 ()`
- `static __forceinline__ __device__ unsigned int optixGetPayload_1 ()`

- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_2 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_3 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_4 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_5 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_6 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_7 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_8 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_9 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_10 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_11 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_12 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_13 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_14 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_15 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_16 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_17 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_18 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_19 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_20 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_21 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_22 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_23 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_24 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_25 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_26 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_27 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_28 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_29 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_30 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetPayload\\_31 \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixSetPayloadTypes](#) (unsigned int typeMask)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixUndefinedValue \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 [optixGetWorldRayOrigin \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 [optixGetWorldRayDirection \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 [optixGetObjectRayOrigin \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 [optixGetObjectRayDirection \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ float [optixGetRayTmin \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ float [optixGetRayTmax \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ float [optixGetRayTime \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetRayFlags \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int [optixGetRayVisibilityMask \(\)](#)
- static \_\_forceinline\_\_ \_\_device\_\_ [OptixTraversableHandle](#) [optixGetInstanceTraversableFromIAS](#) ([OptixTraversableHandle](#) ias, unsigned int instIdx)
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixGetTriangleVertexData](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixGetMicroTriangleVertexData](#) (float3 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixGetMicroTriangleBarycentricsData](#) (float2 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void [optixGetLinearCurveVertexData](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])

- static \_\_forceinline\_\_ \_\_device\_\_ void `optixGetQuadraticBSplineVertexData` (`OptixTraversableHandle` gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ void `optixGetCubicBSplineVertexData` (`OptixTraversableHandle` gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static \_\_forceinline\_\_ \_\_device\_\_ void `optixGetCatmullRomVertexData` (`OptixTraversableHandle` gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static \_\_forceinline\_\_ \_\_device\_\_ void `optixGetCubicBezierVertexData` (`OptixTraversableHandle` gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static \_\_forceinline\_\_ \_\_device\_\_ void `optixGetRibbonVertexData` (`OptixTraversableHandle` gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static \_\_forceinline\_\_ \_\_device\_\_ float3 `optixGetRibbonNormal` (`OptixTraversableHandle` gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static \_\_forceinline\_\_ \_\_device\_\_ void `optixGetSphereData` (`OptixTraversableHandle` gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static \_\_forceinline\_\_ \_\_device\_\_ `OptixTraversableHandle` `optixGetGASTraversableHandle` ()
- static \_\_forceinline\_\_ \_\_device\_\_ float `optixGetGASMotionTimeBegin` (`OptixTraversableHandle` gas)
- static \_\_forceinline\_\_ \_\_device\_\_ float `optixGetGASMotionTimeEnd` (`OptixTraversableHandle` gas)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int `optixGetGASMotionStepCount` (`OptixTraversableHandle` gas)
- static \_\_forceinline\_\_ \_\_device\_\_ void `optixGetWorldToObjectTransformMatrix` (float m[12])
- static \_\_forceinline\_\_ \_\_device\_\_ void `optixGetObjectToWorldTransformMatrix` (float m[12])
- static \_\_forceinline\_\_ \_\_device\_\_ float3 `optixTransformPointFromWorldToObjectSpace` (float3 point)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 `optixTransformVectorFromWorldToObjectSpace` (float3 vec)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 `optixTransformNormalFromWorldToObjectSpace` (float3 normal)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 `optixTransformPointFromObjectToWorldSpace` (float3 point)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 `optixTransformVectorFromObjectToWorldSpace` (float3 vec)
- static \_\_forceinline\_\_ \_\_device\_\_ float3 `optixTransformNormalFromObjectToWorldSpace` (float3 normal)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int `optixGetTransformListSize` ()
- static \_\_forceinline\_\_ \_\_device\_\_ `OptixTraversableHandle` `optixGetTransformListHandle` (unsigned int index)
- static \_\_forceinline\_\_ \_\_device\_\_ `OptixTransformType` `optixGetTransformTypeFromHandle` (`OptixTraversableHandle` handle)
- static \_\_forceinline\_\_ \_\_device\_\_ const `OptixStaticTransform` \* `optixGetStaticTransformFromHandle` (`OptixTraversableHandle` handle)
- static \_\_forceinline\_\_ \_\_device\_\_ const `OptixSRTMotionTransform` \* `optixGetSRTMotionTransformFromHandle` (`OptixTraversableHandle` handle)
- static \_\_forceinline\_\_ \_\_device\_\_ const `OptixMatrixMotionTransform` \* `optixGetMatrixMotionTransformFromHandle` (`OptixTraversableHandle` handle)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int `optixGetInstanceIdFromHandle` (`OptixTraversableHandle` handle)
- static \_\_forceinline\_\_ \_\_device\_\_ `OptixTraversableHandle` `optixGetInstanceChildFromHandle` (`OptixTraversableHandle` handle)

- static \_\_forceinline\_\_ \_\_device\_\_ const float4 \* optixGetInstanceTransformFromHandle (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ const float4 \* optixGetInstanceInverseTransformFromHandle (OptixTraversableHandle handle)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6, unsigned int a7)
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_0 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_1 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_2 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_3 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_4 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_5 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_6 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetAttribute\_7 ()
- static \_\_forceinline\_\_ \_\_device\_\_ void optixTerminateRay ()
- static \_\_forceinline\_\_ \_\_device\_\_ void optixIgnoreIntersection ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPrimitiveIndex ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetSbtGASIndex ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetInstanceId ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetInstanceIndex ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetHitKind ()
- static \_\_forceinline\_\_ \_\_device\_\_ OptixPrimitiveType optixGetPrimitiveType (unsigned int hitKind)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsFrontFaceHit (unsigned int hitKind)
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsBackFaceHit (unsigned int hitKind)
- static \_\_forceinline\_\_ \_\_device\_\_ OptixPrimitiveType optixGetPrimitiveType ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsFrontFaceHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsBackFaceHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsTriangleHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsTriangleFrontFaceHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsTriangleBackFaceHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsDisplacedMicromeshTriangleHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsDisplacedMicromeshTriangleFrontFaceHit ()

- static \_\_forceinline\_\_ \_\_device\_\_ bool optixIsDisplacedMicromeshTriangleBackFaceHit ()
- static \_\_forceinline\_\_ \_\_device\_\_ float2 optixGetTriangleBarycentrics ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetCurveParameter ()
- static \_\_forceinline\_\_ \_\_device\_\_ float2 optixGetRibbonParameters ()
- static \_\_forceinline\_\_ \_\_device\_\_ uint3 optixGetLaunchIndex ()
- static \_\_forceinline\_\_ \_\_device\_\_ uint3 optixGetLaunchDimensions ()
- static \_\_forceinline\_\_ \_\_device\_\_ CUdeviceptr optixGetSbtDataPointer ()
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6)
- static \_\_forceinline\_\_ \_\_device\_\_ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6, unsigned int exceptionDetail7)
- static \_\_forceinline\_\_ \_\_device\_\_ int optixGetExceptionCode ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_0 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_1 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_2 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_3 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_4 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_5 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_6 ()
- static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetExceptionDetail\_7 ()
- static \_\_forceinline\_\_ \_\_device\_\_ OptixTraversableHandle optixGetExceptionInvalidTraversable ()
- static \_\_forceinline\_\_ \_\_device\_\_ int optixGetExceptionInvalidSbtOffset ()
- static \_\_forceinline\_\_ \_\_device\_\_ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay ()
- static \_\_forceinline\_\_ \_\_device\_\_ OptixParameterMismatchExceptionDetails optixGetExceptionParameterMismatch ()
- static \_\_forceinline\_\_ \_\_device\_\_ char \* optixGetExceptionLineInfo ()
- template<typename ReturnT, typename... ArgTypes>  
static \_\_forceinline\_\_ \_\_device\_\_ ReturnT optixDirectCall (unsigned int sbtIndex, ArgTypes... args)



- `template<typename ReturnT, typename... ArgTypes>`  
`static __forceinline__ __device__ ReturnT optixContinuationCall (unsigned int sbtIndex,`  
`ArgTypes... args)`
- `static __forceinline__ __device__ uint4 optixTexFootprint2D (unsigned long long tex, unsigned`  
`int texInfo, float x, float y, unsigned int *singleMipLevel)`
- `static __forceinline__ __device__ uint4 optixTexFootprint2DLod (unsigned long long tex,`  
`unsigned int texInfo, float x, float y, float level, bool coarse, unsigned int *singleMipLevel)`
- `static __forceinline__ __device__ uint4 optixTexFootprint2DGrad (unsigned long long tex,`  
`unsigned int texInfo, float x, float y, float dPdx_x, float dPdx_y, float dPdy_x, float dPdy_y, bool`  
`coarse, unsigned int *singleMipLevel)`

### 8.11.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

OptiX public API Reference - Device API declarations

### 8.11.2 Macro Definition Documentation

#### 8.11.2.1 \_\_OPTIX\_INCLUDE\_INTERNAL\_HEADERS\_\_

```
#define __OPTIX_INCLUDE_INTERNAL_HEADERS__
```

## 8.12 optix\_device.h

[Go to the documentation of this file.](#)

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
3 *
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly
8 * prohibited.
9 *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE. IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
26
27 #ifndef OPTIX_DEVICE_H
28 #define OPTIX_DEVICE_H
29
30 #if defined(__cplusplus) && (__cplusplus < 201103L) && !defined(_WIN32)
31 #error Device code for OptiX requires at least C++11. Consider adding "--std c++11" to the nvcc
command-line.
32 #endif
33
34 #include "optix_types.h"
```

```

35
38
58 template <typename... Payload>
59 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
60  float3          rayOrigin,
61  float3          rayDirection,
62  float          tmin,
63  float          tmax,
64  float          rayTime,
65  OptixVisibilityMask visibilityMask,
66  unsigned int    rayFlags,
67  unsigned int    SBTOffset,
68  unsigned int    SBTstride,
69  unsigned int    missSBTIndex,
70  Payload&...     payload);
71
72
88 template <typename... Payload>
89 static __forceinline__ __device__ void optixTrace(OptixPayloadTypeID type,
90  OptixTraversableHandle handle,
91  float3          rayOrigin,
92  float3          rayDirection,
93  float          tmin,
94  float          tmax,
95  float          rayTime,
96  OptixVisibilityMask visibilityMask,
97  unsigned int    rayFlags,
98  unsigned int    SBTOffset,
99  unsigned int    SBTstride,
100  unsigned int    missSBTIndex,
101  Payload&...     payload);
102
103
105 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p);
107 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p);
109 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p);
111 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p);
113 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p);
115 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p);
117 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p);
119 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p);
120
122 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p);
124 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p);
126 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p);
128 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p);
130 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p);
132 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p);
134 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p);
136 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p);
138 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p);
140 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p);
142 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p);
144 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p);
146 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p);
148 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p);
150 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p);
152 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p);
154 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p);
156 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p);
158 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p);
160 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p);
162 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p);
164 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p);
166 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p);
168 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p);
169

```

```

171 static __forceinline__ __device__ unsigned int optixGetPayload_0();
173 static __forceinline__ __device__ unsigned int optixGetPayload_1();
175 static __forceinline__ __device__ unsigned int optixGetPayload_2();
177 static __forceinline__ __device__ unsigned int optixGetPayload_3();
179 static __forceinline__ __device__ unsigned int optixGetPayload_4();
181 static __forceinline__ __device__ unsigned int optixGetPayload_5();
183 static __forceinline__ __device__ unsigned int optixGetPayload_6();
185 static __forceinline__ __device__ unsigned int optixGetPayload_7();
186
188 static __forceinline__ __device__ unsigned int optixGetPayload_8();
190 static __forceinline__ __device__ unsigned int optixGetPayload_9();
192 static __forceinline__ __device__ unsigned int optixGetPayload_10();
194 static __forceinline__ __device__ unsigned int optixGetPayload_11();
196 static __forceinline__ __device__ unsigned int optixGetPayload_12();
198 static __forceinline__ __device__ unsigned int optixGetPayload_13();
200 static __forceinline__ __device__ unsigned int optixGetPayload_14();
202 static __forceinline__ __device__ unsigned int optixGetPayload_15();
204 static __forceinline__ __device__ unsigned int optixGetPayload_16();
206 static __forceinline__ __device__ unsigned int optixGetPayload_17();
208 static __forceinline__ __device__ unsigned int optixGetPayload_18();
210 static __forceinline__ __device__ unsigned int optixGetPayload_19();
212 static __forceinline__ __device__ unsigned int optixGetPayload_20();
214 static __forceinline__ __device__ unsigned int optixGetPayload_21();
216 static __forceinline__ __device__ unsigned int optixGetPayload_22();
218 static __forceinline__ __device__ unsigned int optixGetPayload_23();
220 static __forceinline__ __device__ unsigned int optixGetPayload_24();
222 static __forceinline__ __device__ unsigned int optixGetPayload_25();
224 static __forceinline__ __device__ unsigned int optixGetPayload_26();
226 static __forceinline__ __device__ unsigned int optixGetPayload_27();
228 static __forceinline__ __device__ unsigned int optixGetPayload_28();
230 static __forceinline__ __device__ unsigned int optixGetPayload_29();
232 static __forceinline__ __device__ unsigned int optixGetPayload_30();
234 static __forceinline__ __device__ unsigned int optixGetPayload_31();
235
242 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int typeMask);
243
245 static __forceinline__ __device__ unsigned int optixUndefinedValue();
246
252 static __forceinline__ __device__ float3 optixGetWorldRayOrigin();
253
259 static __forceinline__ __device__ float3 optixGetWorldRayDirection();
260
264 static __forceinline__ __device__ float3 optixGetObjectRayOrigin();
265
269 static __forceinline__ __device__ float3 optixGetObjectRayDirection();
270
274 static __forceinline__ __device__ float optixGetRayTmin();
275
280 static __forceinline__ __device__ float optixGetRayTmax();
281
286 static __forceinline__ __device__ float optixGetRayTime();
287
291 static __forceinline__ __device__ unsigned int optixGetRayFlags();
292
296 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask();
297
301 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias, unsigned int instIdx);
302
309 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas, unsigned
int primIdx, unsigned int sbtGASIndex, float time, float3 data[3]);
310
313 static __forceinline__ __device__ void optixGetMicroTriangleVertexData(float3 data[3]);
315 static __forceinline__ __device__ void optixGetMicroTriangleBarycentricsData(float2 data[3]);
316
324 static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2]);

```



```

325
333 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3]);
334
342 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4]);
343
351 static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas, unsigned
int primIdx, unsigned int sbtGASIndex, float time, float4 data[4]);
352
360 static __forceinline__ __device__ void optixGetCubicBezierVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4]);
361
369 static __forceinline__ __device__ void optixGetRibbonVertexData(OptixTraversableHandle gas, unsigned int
primIdx, unsigned int sbtGASIndex, float time, float4 data[3]);
370
372 static __forceinline__ __device__ float3 optixGetRibbonNormal(OptixTraversableHandle gas, unsigned int
primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters);
373
380 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas, unsigned int
primIdx, unsigned int sbtGASIndex, float time, float4 data[1]);
381
384 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle();
385
387 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle gas);
388
390 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle gas);
391
393 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle gas);
394
398 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12]);
399
403 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12]);
404
409 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point);
410
415 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec);
416
421 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal);
422
427 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point);
428
433 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec);
434
439 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal);
440
444 static __forceinline__ __device__ unsigned int optixGetTransformListSize();
445
449 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int index);
450
451
453 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle);
454
458 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle);
459
463 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle);
464
468 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle);
469
473 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle);
474
478 static __forceinline__ __device__ OptixTraversableHandle

```

```

optixGetInstanceChildFromHandle(OptixTraversableHandle handle);
479
483 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle);
484
488 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle);
489
506 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind);
507
511 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0);
512
516 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1);
517
521 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1, unsigned int a2);
522
526 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
527  unsigned int hitKind,
528  unsigned int a0,
529  unsigned int a1,
530  unsigned int a2,
531  unsigned int a3);
532
536 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
537  unsigned int hitKind,
538  unsigned int a0,
539  unsigned int a1,
540  unsigned int a2,
541  unsigned int a3,
542  unsigned int a4);
543
547 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
548  unsigned int hitKind,
549  unsigned int a0,
550  unsigned int a1,
551  unsigned int a2,
552  unsigned int a3,
553  unsigned int a4,
554  unsigned int a5);
555
559 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
560  unsigned int hitKind,
561  unsigned int a0,
562  unsigned int a1,
563  unsigned int a2,
564  unsigned int a3,
565  unsigned int a4,
566  unsigned int a5,
567  unsigned int a6);
568
572 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
573  unsigned int hitKind,
574  unsigned int a0,
575  unsigned int a1,
576  unsigned int a2,
577  unsigned int a3,
578  unsigned int a4,
579  unsigned int a5,
580  unsigned int a6,
581  unsigned int a7);
582
584 static __forceinline__ __device__ unsigned int optixGetAttribute_0();
586 static __forceinline__ __device__ unsigned int optixGetAttribute_1();
588 static __forceinline__ __device__ unsigned int optixGetAttribute_2();

```

```

590 static __forceinline__ __device__ unsigned int optixGetAttribute_3();
592 static __forceinline__ __device__ unsigned int optixGetAttribute_4();
594 static __forceinline__ __device__ unsigned int optixGetAttribute_5();
596 static __forceinline__ __device__ unsigned int optixGetAttribute_6();
598 static __forceinline__ __device__ unsigned int optixGetAttribute_7();
599
603 static __forceinline__ __device__ void optixTerminateRay();
604
608 static __forceinline__ __device__ void optixIgnoreIntersection();
609
610
622 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex();
623
629 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex();
630
631
639 static __forceinline__ __device__ unsigned int optixGetInstanceId();
640
646 static __forceinline__ __device__ unsigned int optixGetInstanceIndex();
647
656 static __forceinline__ __device__ unsigned int optixGetHitKind();
657
659 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind);
660
662 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind);
663
665 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind);
666
668 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType();
669
671 static __forceinline__ __device__ bool optixIsFrontFaceHit();
672
674 static __forceinline__ __device__ bool optixIsBackFaceHit();
675
677 static __forceinline__ __device__ bool optixIsTriangleHit();
678
680 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit();
681
683 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit();
684
686 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit();
687
689 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleFrontFaceHit();
690
692 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleBackFaceHit();
693
698 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics();
699
702 static __forceinline__ __device__ float optixGetCurveParameter();
703
706 static __forceinline__ __device__ float2 optixGetRibbonParameters();
707
711 static __forceinline__ __device__ uint3 optixGetLaunchIndex();
712
714 static __forceinline__ __device__ uint3 optixGetLaunchDimensions();
715
717 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer();
718
730 static __forceinline__ __device__ void optixThrowException(int exceptionCode);
731
735 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0);
736
740 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
741                                     unsigned int exceptionDetail0,
742                                     unsigned int exceptionDetail1);
743

```

```

747 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
748   unsigned int exceptionDetail0,
749   unsigned int exceptionDetail1,
750   unsigned int exceptionDetail2);
751
755 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
756   unsigned int exceptionDetail0,
757   unsigned int exceptionDetail1,
758   unsigned int exceptionDetail2,
759   unsigned int exceptionDetail3);
760
764 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
765   unsigned int exceptionDetail0,
766   unsigned int exceptionDetail1,
767   unsigned int exceptionDetail2,
768   unsigned int exceptionDetail3,
769   unsigned int exceptionDetail4);
770
774 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
775   unsigned int exceptionDetail0,
776   unsigned int exceptionDetail1,
777   unsigned int exceptionDetail2,
778   unsigned int exceptionDetail3,
779   unsigned int exceptionDetail4,
780   unsigned int exceptionDetail5);
781
785 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
786   unsigned int exceptionDetail0,
787   unsigned int exceptionDetail1,
788   unsigned int exceptionDetail2,
789   unsigned int exceptionDetail3,
790   unsigned int exceptionDetail4,
791   unsigned int exceptionDetail5,
792   unsigned int exceptionDetail6);
793
797 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
798   unsigned int exceptionDetail0,
799   unsigned int exceptionDetail1,
800   unsigned int exceptionDetail2,
801   unsigned int exceptionDetail3,
802   unsigned int exceptionDetail4,
803   unsigned int exceptionDetail5,
804   unsigned int exceptionDetail6,
805   unsigned int exceptionDetail7);
806
810 static __forceinline__ __device__ int optixGetExceptionCode();
811
818 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0();
819
823 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1();
824
828 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2();
829
833 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3();
834
838 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4();
839
843 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5();
844
848 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6();
849
853 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7();
854
860 static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable();
861
867 static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset();
868

```

```

877 static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay();
878
891 static __forceinline__ __device__ OptixParameterMismatchExceptionDetails
optixGetExceptionParameterMismatch();
892
903 static __forceinline__ __device__ char* optixGetExceptionLineInfo();
904
919 template <typename ReturnT, typename... ArgTypes>
920 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args);
921
922
938 template <typename ReturnT, typename... ArgTypes>
939 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes... args);
940
941
1004 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int
texInfo, float x, float y, unsigned int* singleMipLevel);
1005
1015 static __forceinline__ __device__ uint4
1016 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool
coarse, unsigned int* singleMipLevel);
1017
1030 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
1031                                unsigned int      texInfo,
1032                                float              x,
1033                                float              y,
1034                                float              dPdx_x,
1035                                float              dPdx_y,
1036                                float              dPdy_x,
1037                                float              dPdy_y,
1038                                bool               coarse,
1039                                unsigned int*      singleMipLevel);
1040 // end group optix_device_api
1042
1043 #define __OPTIX_INCLUDE_INTERNAL_HEADERS__
1044
1045 #include "internal/optix_device_impl.h"
1046
1047 #endif // OPTIX_DEVICE_H

```

## 8.13 optix\_function\_table.h File Reference

### Classes

- struct [OptixFunctionTable](#)

### Macros

- #define [OPTIX\\_ABI\\_VERSION](#) 84

### Typedefs

- typedef struct [OptixFunctionTable](#) [OptixFunctionTable](#)

### 8.13.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

## 8.13.2 Macro Definition Documentation

### 8.13.2.1 OPTIX\_ABI\_VERSION

```
#define OPTIX_ABI_VERSION 84
```

The OptiX ABI version.

## 8.14 optix\_function\_table.h

[Go to the documentation of this file.](#)

```
1 /*
2  * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
3  *
4  * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5  * rights in and to this software, related documentation and any modifications thereto.
6  * Any use, reproduction, disclosure or distribution of this software and related
7  * documentation without an express license agreement from NVIDIA Corporation is strictly
8  * prohibited.
9  *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE. IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
24
25 #ifndef OPTIX_OPTIX_FUNCTION_TABLE_H
26 #define OPTIX_OPTIX_FUNCTION_TABLE_H
27
29 #define OPTIX_ABI_VERSION 84
30
31 #ifndef OPTIX_DEFINE_ABI_VERSION_ONLY
32
33 #include "optix_types.h"
34
35 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
36 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver types must be defined through other
37 // means before including optix headers.
38 #include <cuda.h>
39 #endif
40
41 #ifdef __cplusplus
42 extern "C" {
43 #endif
44
47
55 typedef struct OptixFunctionTable
56 {
57     /*@ {
58
59
61     const char* (*optixGetErrorName)(OptixResult result);
62
64     const char* (*optixGetErrorString)(OptixResult result);
65
66     /*@ }
```

```

68     ///  

69  

71     OptixResult (*optixDeviceContextCreate)(CUcontext fromContext, const OptixDeviceContextOptions*  

options, OptixDeviceContext* context);  

72  

74     OptixResult (*optixDeviceContextDestroy)(OptixDeviceContext context);  

75  

77     OptixResult (*optixDeviceContextGetProperty)(OptixDeviceContext context, OptixDeviceProperty  

property, void* value, size_t sizeInBytes);  

78  

80     OptixResult (*optixDeviceContextSetLogCallback)(OptixDeviceContext context,  

81   OptixLogCallback callbackFunction,  

82   void* callbackData,  

83   unsigned int callbackLevel);  

84  

86     OptixResult (*optixDeviceContextSetCacheEnabled)(OptixDeviceContext context, int enabled);  

87  

89     OptixResult (*optixDeviceContextSetCacheLocation)(OptixDeviceContext context, const char* location);  

90  

92     OptixResult (*optixDeviceContextSetCacheDatabaseSizes)(OptixDeviceContext context, size_t  

lowWaterMark, size_t highWaterMark);  

93  

95     OptixResult (*optixDeviceContextGetCacheEnabled)(OptixDeviceContext context, int* enabled);  

96  

98     OptixResult (*optixDeviceContextGetCacheLocation)(OptixDeviceContext context, char* location, size_t  

locationSize);  

99  

101    OptixResult (*optixDeviceContextGetCacheDatabaseSizes)(OptixDeviceContext context, size_t*  

lowWaterMark, size_t* highWaterMark);  

102  

103    ///  

105    ///  

106  

108    OptixResult (*optixModuleCreate)(OptixDeviceContext context,  

109                                    const OptixModuleCompileOptions* moduleCompileOptions,  

110                                    const OptixPipelineCompileOptions* pipelineCompileOptions,  

111                                    const char* input,  

112                                    size_t inputSize,  

113                                    char* logString,  

114                                    size_t* logStringSize,  

115                                    OptixModule* module);  

116  

118    OptixResult (*optixModuleCreateWithTasks)(OptixDeviceContext context,  

119  const OptixModuleCompileOptions* moduleCompileOptions,  

120  const OptixPipelineCompileOptions* pipelineCompileOptions,  

121  const char* input,  

122  size_t inputSize,  

123  char* logString,  

124  size_t* logStringSize,  

125  OptixModule* module,  

126  OptixTask* firstTask);  

127  

129    OptixResult (*optixModuleGetCompilationState)(OptixModule module, OptixModuleCompileState* state);  

130  

132    OptixResult (*optixModuleDestroy)(OptixModule module);  

133  

135    OptixResult(*optixBuiltinISModuleGet)(OptixDeviceContext context,  

136   const OptixModuleCompileOptions* moduleCompileOptions,  

137   const OptixPipelineCompileOptions* pipelineCompileOptions,  

138   const OptixBuiltinISOptions* builtinISOptions,  

139   OptixModule* builtinModule);  

140  

141    ///  

143    ///  

144  

146    OptixResult (*optixTaskExecute)(OptixTask task,  

147                                    OptixTask* additionalTasks,

```



```

148         unsigned int    maxNumAdditionalTasks,
149         unsigned int*    numAdditionalTasksCreated);
150     //@ }
151     //@ {
152
153     OptixResult (*optixProgramGroupCreate)(OptixDeviceContext    context,
154         const OptixProgramGroupDesc*    programDescriptions,
155         unsigned int                    numProgramGroups,
156         const OptixProgramGroupOptions* options,
157         char*                          logString,
158         size_t*                        logStringSize,
159         OptixProgramGroup*             programGroups);
160
161     OptixResult (*optixProgramGroupDestroy)(OptixProgramGroup programGroup);
162
163     OptixResult (*optixProgramGroupGetStackSize)(OptixProgramGroup programGroup, OptixStackSizes*
164     stackSizes, OptixPipeline pipeline);
165
166     //@ }
167     //@ {
168
169     OptixResult (*optixPipelineCreate)(OptixDeviceContext    context,
170         const OptixPipelineCompileOptions* pipelineCompileOptions,
171         const OptixPipelineLinkOptions*    pipelineLinkOptions,
172         const OptixProgramGroup*          programGroups,
173         unsigned int                      numProgramGroups,
174         char*                            logString,
175         size_t*                          logStringSize,
176         OptixPipeline*                   pipeline);
177
178     OptixResult (*optixPipelineDestroy)(OptixPipeline pipeline);
179
180     OptixResult (*optixPipelineSetStackSize)(OptixPipeline pipeline,
181         unsigned int    directCallableStackSizeFromTraversal,
182         unsigned int    directCallableStackSizeFromState,
183         unsigned int    continuationStackSize,
184         unsigned int    maxTraversableGraphDepth);
185
186     //@ }
187     //@ {
188
189     OptixResult (*optixAccelComputeMemoryUsage)(OptixDeviceContext    context,
190         const OptixAccelBuildOptions* accelOptions,
191         const OptixBuildInput*        buildInputs,
192         unsigned int                  numBuildInputs,
193         OptixAccelBufferSizes*        bufferSizes);
194
195     OptixResult (*optixAccelBuild)(OptixDeviceContext    context,
196         CUstream    stream,
197         const OptixAccelBuildOptions* accelOptions,
198         const OptixBuildInput*        buildInputs,
199         unsigned int                  numBuildInputs,
200         CUdeviceptr    tempBuffer,
201         size_t          tempBufferSizeInBytes,
202         CUdeviceptr    outputBuffer,
203         size_t          outputBufferSizeInBytes,
204         OptixTraversableHandle*    outputHandle,
205         const OptixAccelEmitDesc*    emittedProperties,
206         unsigned int                  numEmittedProperties);
207
208     OptixResult (*optixAccelGetRelocationInfo)(OptixDeviceContext context, OptixTraversableHandle
209     handle, OptixRelocationInfo* info);
210
211     OptixResult (*optixCheckRelocationCompatibility)(OptixDeviceContext    context,
212         const OptixRelocationInfo* info,
213         int*                        compatible);
214

```



```

226
228     OptixResult (*optixAccelRelocate)(OptixDeviceContext      context,
229                                     CUstream                stream,
230                                     const OptixRelocationInfo* info,
231                                     const OptixRelocateInput*  relocateInputs,
232                                     size_t                    numRelocateInputs,
233                                     CUdeviceptr               targetAccel,
234                                     size_t                    targetAccelSizeInBytes,
235                                     OptixTraversableHandle*    targetHandle);
236
237
239     OptixResult (*optixAccelCompact)(OptixDeviceContext      context,
240                                     CUstream                stream,
241                                     OptixTraversableHandle    inputHandle,
242                                     CUdeviceptr               outputBuffer,
243                                     size_t                    outputBufferSizeInBytes,
244                                     OptixTraversableHandle*    outputHandle);
245
246     OptixResult (*optixAccelEmitProperty)(OptixDeviceContext      context,
247   CUstream                stream,
248   OptixTraversableHandle    handle,
249   const OptixAccelEmitDesc* emittedProperty);
250
252     OptixResult (*optixConvertPointerToTraversableHandle)(OptixDeviceContext      onDevice,
253   CUdeviceptr                pointer,
254   OptixTraversableType    traversableType,
255   OptixTraversableHandle*    traversableHandle);
256
258     OptixResult (*optixOpacityMicromapArrayComputeMemoryUsage)(OptixDeviceContext
context,
259  const OptixOpacityMicromapArrayBuildInput*
buildInput,
260  OptixMicromapBufferSizes*
bufferSizes);
261
263     OptixResult (*optixOpacityMicromapArrayBuild)(OptixDeviceContext      context,
264  CUstream                stream,
265  const OptixOpacityMicromapArrayBuildInput* buildInput,
266  const OptixMicromapBuffers*    buffers);
267
269     OptixResult (*optixOpacityMicromapArrayGetRelocationInfo)(OptixDeviceContext      context,
270  CUdeviceptr                opacityMicromapArray,
271  OptixRelocationInfo*    info);
272
274     OptixResult (*optixOpacityMicromapArrayRelocate)(OptixDeviceContext      context,
275   CUstream                stream,
276   const OptixRelocationInfo* info,
277   CUdeviceptr                targetOpacityMicromapArray,
278   size_t                    targetOpacityMicromapArraySizeInBytes);
279
281     OptixResult (*optixDisplacementMicromapArrayComputeMemoryUsage)(OptixDeviceContext context,
282  const
OptixDisplacementMicromapArrayBuildInput* buildInput,
283  OptixMicromapBufferSizes* bufferSizes);
284
286     OptixResult (*optixDisplacementMicromapArrayBuild)(OptixDeviceContext
context,
287   CUstream                stream,
288   const OptixDisplacementMicromapArrayBuildInput*
buildInput,
289   const OptixMicromapBuffers*
buffers);
290
291     //@ }
293     //@ {
294

```

```

296     OptixResult (*optixSbtRecordPackHeader)(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer);
297
299     OptixResult (*optixLaunch)(OptixPipeline          pipeline,
300                               CUstream                stream,
301                               CUdeviceptr             pipelineParams,
302                               size_t                  pipelineParamsSize,
303                               const OptixShaderBindingTable* sbt,
304                               unsigned int            width,
305                               unsigned int            height,
306                               unsigned int            depth);
307
308     //@ }
309     //@ {
310
311
313     OptixResult (*optixDenoiserCreate)(OptixDeviceContext context, OptixDenoiserModelKind modelKind,
const OptixDenoiserOptions* options, OptixDenoiser* returnHandle);
314
316     OptixResult (*optixDenoiserDestroy)(OptixDenoiser handle);
317
319     OptixResult (*optixDenoiserComputeMemoryResources)(const OptixDenoiser handle,
320   unsigned int            maximumInputWidth,
321   unsigned int            maximumInputHeight,
322   OptixDenoiserSizes* returnSizes);
323
325     OptixResult (*optixDenoiserSetup)(OptixDenoiser denoiser,
326                                       CUstream        stream,
327                                       unsigned int     inputWidth,
328                                       unsigned int     inputHeight,
329                                       CUdeviceptr      state,
330                                       size_t           stateSizeInBytes,
331                                       CUdeviceptr      scratch,
332                                       size_t           scratchSizeInBytes);
333
335     OptixResult (*optixDenoiserInvoke)(OptixDenoiser          denoiser,
336                                       CUstream                stream,
337                                       const OptixDenoiserParams* params,
338                                       CUdeviceptr             denoiserState,
339                                       size_t                   denoiserStateSizeInBytes,
340                                       const OptixDenoiserGuideLayer * guideLayer,
341                                       const OptixDenoiserLayer *   layers,
342                                       unsigned int              numLayers,
343                                       unsigned int              inputOffsetX,
344                                       unsigned int              inputOffsetY,
345                                       CUdeviceptr              scratch,
346                                       size_t                   scratchSizeInBytes);
347
349     OptixResult (*optixDenoiserComputeIntensity)(OptixDenoiser          handle,
350  CUstream                stream,
351  const OptixImage2D* inputImage,
352  CUdeviceptr             outputIntensity,
353  CUdeviceptr             scratch,
354  size_t                   scratchSizeInBytes);
355
357     OptixResult (*optixDenoiserComputeAverageColor)(OptixDenoiser          handle,
358   CUstream                stream,
359   const OptixImage2D* inputImage,
360   CUdeviceptr             outputAverageColor,
361   CUdeviceptr             scratch,
362   size_t                   scratchSizeInBytes);
363
365     OptixResult (*optixDenoiserCreateWithUserModel)(OptixDeviceContext context, const void * data, size_t
dataSizeInBytes, OptixDenoiser* returnHandle);
366     //@ }
367
368 } OptixFunctionTable;
369 // end group optix_function_table

```

```

371
372 #ifdef __cplusplus
373 }
374 #endif
375
376 #endif /* OPTIX_DEFINE_ABI_VERSION_ONLY */
377
378 #endif /* OPTIX_OPTIX_FUNCTION_TABLE_H */

```

## 8.15 optix\_function\_table\_definition.h File Reference

### Variables

- [OptixFunctionTable g\\_optixFunctionTable](#)

### 8.15.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

## 8.16 optix\_function\_table\_definition.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
3  *
4  * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5  * rights in and to this software, related documentation and any modifications thereto.
6  * Any use, reproduction, disclosure or distribution of this software and related
7  * documentation without an express license agreement from NVIDIA Corporation is strictly
8  * prohibited.
9  *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE. IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
24
25 #ifndef OPTIX_OPTIX_FUNCTION_TABLE_DEFINITION_H
26 #define OPTIX_OPTIX_FUNCTION_TABLE_DEFINITION_H
27
28 #include "optix_function_table.h"
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
34 OptixFunctionTable g_optixFunctionTable;
35 // end group optix_function_table
36
37 #ifdef __cplusplus
38 }
39 #endif
40
41 #endif // OPTIX_OPTIX_FUNCTION_TABLE_DEFINITION_H

```

## 8.17 optix\_host.h File Reference

### Functions

- `const char * optixGetErrorName (OptixResult result)`
- `const char * optixGetErrorString (OptixResult result)`
- `OptixResult optixDeviceContextCreate (CUcontext fromContext, const OptixDeviceContextOptions *options, OptixDeviceContext *context)`
- `OptixResult optixDeviceContextDestroy (OptixDeviceContext context)`
- `OptixResult optixDeviceContextGetProperty (OptixDeviceContext context, OptixDeviceProperty property, void *value, size_t sizeInBytes)`
- `OptixResult optixDeviceContextSetLogCallback (OptixDeviceContext context, OptixLogCallback callbackFunction, void *callbackData, unsigned int callbackLevel)`
- `OptixResult optixDeviceContextSetCacheEnabled (OptixDeviceContext context, int enabled)`
- `OptixResult optixDeviceContextSetCacheLocation (OptixDeviceContext context, const char *location)`
- `OptixResult optixDeviceContextSetCacheDatabaseSizes (OptixDeviceContext context, size_t lowWaterMark, size_t highWaterMark)`
- `OptixResult optixDeviceContextGetCacheEnabled (OptixDeviceContext context, int *enabled)`
- `OptixResult optixDeviceContextGetCacheLocation (OptixDeviceContext context, char *location, size_t locationSize)`
- `OptixResult optixDeviceContextGetCacheDatabaseSizes (OptixDeviceContext context, size_t *lowWaterMark, size_t *highWaterMark)`
- `OptixResult optixPipelineCreate (OptixDeviceContext context, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixPipelineLinkOptions *pipelineLinkOptions, const OptixProgramGroup *programGroups, unsigned int numProgramGroups, char *logString, size_t *logStringSize, OptixPipeline *pipeline)`
- `OptixResult optixPipelineDestroy (OptixPipeline pipeline)`
- `OptixResult optixPipelineSetStackSize (OptixPipeline pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)`
- `OptixResult optixModuleCreate (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module)`
- `OptixResult optixModuleCreateWithTasks (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module, OptixTask *firstTask)`
- `OptixResult optixModuleGetCompilationState (OptixModule module, OptixModuleCompileState *state)`
- `OptixResult optixModuleDestroy (OptixModule module)`
- `OptixResult optixBuiltinISModuleGet (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixBuiltinISOOptions *builtinISOOptions, OptixModule *builtinModule)`
- `OptixResult optixTaskExecute (OptixTask task, OptixTask *additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int *numAdditionalTasksCreated)`
- `OptixResult optixProgramGroupGetStackSize (OptixProgramGroup programGroup, OptixStackSizes *stackSizes, OptixPipeline pipeline)`
- `OptixResult optixProgramGroupCreate (OptixDeviceContext context, const OptixProgramGroupDesc *programDescriptions, unsigned int numProgramGroups, const`

- OptixProgramGroupOptions \*options, char \*logString, size\_t \*logStringSize,  
OptixProgramGroup \*programGroups)
- OptixResult optixProgramGroupDestroy (OptixProgramGroup programGroup)
  - OptixResult optixLaunch (OptixPipeline pipeline, CUstream stream, CUdeviceptr pipelineParams, size\_t pipelineParamsSize, const OptixShaderBindingTable \*sbt, unsigned int width, unsigned int height, unsigned int depth)
  - OptixResult optixSbtRecordPackHeader (OptixProgramGroup programGroup, void \*sbtRecordHeaderHostPointer)
  - OptixResult optixAccelComputeMemoryUsage (OptixDeviceContext context, const OptixAccelBuildOptions \*accelOptions, const OptixBuildInput \*buildInputs, unsigned int numBuildInputs, OptixAccelBufferSizes \*bufferSizes)
  - OptixResult optixAccelBuild (OptixDeviceContext context, CUstream stream, const OptixAccelBuildOptions \*accelOptions, const OptixBuildInput \*buildInputs, unsigned int numBuildInputs, CUdeviceptr tempBuffer, size\_t tempBufferSizeInBytes, CUdeviceptr outputBuffer, size\_t outputBufferSizeInBytes, OptixTraversableHandle \*outputHandle, const OptixAccelEmitDesc \*emittedProperties, unsigned int numEmittedProperties)
  - OptixResult optixAccelGetRelocationInfo (OptixDeviceContext context, OptixTraversableHandle handle, OptixRelocationInfo \*info)
  - OptixResult optixCheckRelocationCompatibility (OptixDeviceContext context, const OptixRelocationInfo \*info, int \*compatible)
  - OptixResult optixAccelRelocate (OptixDeviceContext context, CUstream stream, const OptixRelocationInfo \*info, const OptixRelocateInput \*relocateInputs, size\_t numRelocateInputs, CUdeviceptr targetAccel, size\_t targetAccelSizeInBytes, OptixTraversableHandle \*targetHandle)
  - OptixResult optixAccelCompact (OptixDeviceContext context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr outputBuffer, size\_t outputBufferSizeInBytes, OptixTraversableHandle \*outputHandle)
  - OptixResult optixAccelEmitProperty (OptixDeviceContext context, CUstream stream, OptixTraversableHandle handle, const OptixAccelEmitDesc \*emittedProperty)
  - OptixResult optixConvertPointerToTraversableHandle (OptixDeviceContext onDevice, CUdeviceptr pointer, OptixTraversableType traversableType, OptixTraversableHandle \*traversableHandle)
  - OptixResult optixOpacityMicromapArrayComputeMemoryUsage (OptixDeviceContext context, const OptixOpacityMicromapArrayBuildInput \*buildInput, OptixMicromapBufferSizes \*bufferSizes)
  - OptixResult optixOpacityMicromapArrayBuild (OptixDeviceContext context, CUstream stream, const OptixOpacityMicromapArrayBuildInput \*buildInput, const OptixMicromapBuffers \*buffers)
  - OptixResult optixOpacityMicromapArrayGetRelocationInfo (OptixDeviceContext context, CUdeviceptr opacityMicromapArray, OptixRelocationInfo \*info)
  - OptixResult optixOpacityMicromapArrayRelocate (OptixDeviceContext context, CUstream stream, const OptixRelocationInfo \*info, CUdeviceptr targetOpacityMicromapArray, size\_t targetOpacityMicromapArraySizeInBytes)
  - OptixResult optixDisplacementMicromapArrayComputeMemoryUsage (OptixDeviceContext context, const OptixDisplacementMicromapArrayBuildInput \*buildInput, OptixMicromapBufferSizes \*bufferSizes)
  - OptixResult optixDisplacementMicromapArrayBuild (OptixDeviceContext context, CUstream stream, const OptixDisplacementMicromapArrayBuildInput \*buildInput, const OptixMicromapBuffers \*buffers)
  - OptixResult optixDenoiserCreate (OptixDeviceContext context, OptixDenoiserModelKind modelKind, const OptixDenoiserOptions \*options, OptixDenoiser \*denoiser)
  - OptixResult optixDenoiserCreateWithUserModel (OptixDeviceContext context, const void \*userData, size\_t userDataSizeInBytes, OptixDenoiser \*denoiser)

- `OptixResult optixDenoiserDestroy` (`OptixDenoiser` denoiser)
- `OptixResult optixDenoiserComputeMemoryResources` (const `OptixDenoiser` denoiser, unsigned int outputWidth, unsigned int outputHeight, `OptixDenoiserSizes` \*returnSizes)
- `OptixResult optixDenoiserSetup` (`OptixDenoiser` denoiser, `CUstream` stream, unsigned int inputWidth, unsigned int inputHeight, `CUdeviceptr` denoiserState, size\_t denoiserStateSizeInBytes, `CUdeviceptr` scratch, size\_t scratchSizeInBytes)
- `OptixResult optixDenoiserInvoke` (`OptixDenoiser` denoiser, `CUstream` stream, const `OptixDenoiserParams` \*params, `CUdeviceptr` denoiserState, size\_t denoiserStateSizeInBytes, const `OptixDenoiserGuideLayer` \*guideLayer, const `OptixDenoiserLayer` \*layers, unsigned int numLayers, unsigned int inputOffsetX, unsigned int inputOffsetY, `CUdeviceptr` scratch, size\_t scratchSizeInBytes)
- `OptixResult optixDenoiserComputeIntensity` (`OptixDenoiser` denoiser, `CUstream` stream, const `OptixImage2D` \*inputImage, `CUdeviceptr` outputIntensity, `CUdeviceptr` scratch, size\_t scratchSizeInBytes)
- `OptixResult optixDenoiserComputeAverageColor` (`OptixDenoiser` denoiser, `CUstream` stream, const `OptixImage2D` \*inputImage, `CUdeviceptr` outputAverageColor, `CUdeviceptr` scratch, size\_t scratchSizeInBytes)

## 8.17.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

OptiX host include file – includes the host api if compiling host code. For the math library routines include `optix_math.h`

## 8.17.2 Function Documentation

### 8.17.2.1 optixAccelBuild()

```
OptixResult optixAccelBuild (
    OptixDeviceContext context,
    CUstream stream,
    const OptixAccelBuildOptions * accelOptions,
    const OptixBuildInput * buildInputs,
    unsigned int numBuildInputs,
    CUdeviceptr tempBuffer,
    size_t tempBufferSizeInBytes,
    CUdeviceptr outputBuffer,
    size_t outputBufferSizeInBytes,
    OptixTraversableHandle * outputHandle,
    const OptixAccelEmitDesc * emittedProperties,
    unsigned int numEmittedProperties )
```

Parameters

|    |                |  |
|----|----------------|--|
| in | <i>context</i> |  |
|----|----------------|--|



## Parameters

|     |                                |                                                           |
|-----|--------------------------------|-----------------------------------------------------------|
| in  | <i>stream</i>                  |                                                           |
| in  | <i>accelOptions</i>            | accel options                                             |
| in  | <i>buildInputs</i>             | an array of <a href="#">OptixBuildInput</a> objects       |
| in  | <i>numBuildInputs</i>          | must be $\geq 1$ for GAS, and $= 1$ for IAS               |
| in  | <i>tempBuffer</i>              | must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT   |
| in  | <i>tempBufferSizeInBytes</i>   |                                                           |
| in  | <i>outputBuffer</i>            | must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT   |
| in  | <i>outputBufferSizeInBytes</i> |                                                           |
| out | <i>outputHandle</i>            |                                                           |
| in  | <i>emittedProperties</i>       | types of requested properties and output buffers          |
| in  | <i>numEmittedProperties</i>    | number of post-build properties to populate (may be zero) |

## 8.17.2.2 optixAccelCompact()

```
OptixResult optixAccelCompact (
    OptixDeviceContext context,
    CUstream stream,
    OptixTraversableHandle inputHandle,
    CUdeviceptr outputBuffer,
    size_t outputBufferSizeInBytes,
    OptixTraversableHandle * outputHandle )
```

After building an acceleration structure, it can be copied in a compacted form to reduce memory. In order to be compacted, OPTIX\_BUILD\_FLAG\_ALLOW\_COMPACTION must be supplied in [OptixAccelBuildOptions::buildFlags](#) passed to [optixAccelBuild](#).

'outputBuffer' is the pointer to where the compacted acceleration structure will be written. This pointer must be a multiple of OPTIX\_ACCEL\_BUFFER\_BYTE\_ALIGNMENT.

The size of the memory specified in 'outputBufferSizeInBytes' should be at least the value computed using the OPTIX\_PROPERTY\_TYPE\_COMPACTED\_SIZE that was reported during [optixAccelBuild](#).

## Parameters

|     |                                |
|-----|--------------------------------|
| in  | <i>context</i>                 |
| in  | <i>stream</i>                  |
| in  | <i>inputHandle</i>             |
| in  | <i>outputBuffer</i>            |
| in  | <i>outputBufferSizeInBytes</i> |
| out | <i>outputHandle</i>            |

## 8.17.2.3 optixAccelComputeMemoryUsage()

```
OptixResult optixAccelComputeMemoryUsage (
```

```

    OptixDeviceContext context,
    const OptixAccelBuildOptions * accelOptions,
    const OptixBuildInput * buildInputs,
    unsigned int numBuildInputs,
    OptixAccelBufferSizes * bufferSizes )

```

#### Parameters

|     |                       |                                                        |
|-----|-----------------------|--------------------------------------------------------|
| in  | <i>context</i>        |                                                        |
| in  | <i>accelOptions</i>   | options for the accel build                            |
| in  | <i>buildInputs</i>    | an array of <a href="#">OptixBuildInput</a> objects    |
| in  | <i>numBuildInputs</i> | number of elements in buildInputs (must be at least 1) |
| out | <i>bufferSizes</i>    | fills in buffer sizes                                  |

#### 8.17.2.4 optixAccelEmitProperty()

```

OptixResult optixAccelEmitProperty (
    OptixDeviceContext context,
    CUstream stream,
    OptixTraversableHandle handle,
    const OptixAccelEmitDesc * emittedProperty )

```

Emit a single property after an acceleration structure was built. The result buffer of the 'emittedProperty' needs to be large enough to hold the requested property (.).

See also [OptixAccelPropertyType](#)).

#### Parameters

|    |                        |                                              |
|----|------------------------|----------------------------------------------|
| in | <i>context</i>         |                                              |
| in | <i>stream</i>          |                                              |
| in | <i>handle</i>          |                                              |
| in | <i>emittedProperty</i> | type of requested property and output buffer |

#### 8.17.2.5 optixAccelGetRelocationInfo()

```

OptixResult optixAccelGetRelocationInfo (
    OptixDeviceContext context,
    OptixTraversableHandle handle,
    OptixRelocationInfo * info )

```

Obtain relocation information, stored in [OptixRelocationInfo](#), for a given context and acceleration structure's traversable handle.

The relocation information can be passed to [optixCheckRelocationCompatibility](#) to determine if an acceleration structure, referenced by 'handle', can be relocated to a different device's memory space (see [optixCheckRelocationCompatibility](#)).

When used with [optixAccelRelocate](#), it provides data necessary for doing the relocation.

If the acceleration structure data associated with 'handle' is copied multiple times, the same



[OptixRelocationInfo](#) can also be used on all copies.

#### Parameters

|     |                |
|-----|----------------|
| in  | <i>context</i> |
| in  | <i>handle</i>  |
| out | <i>info</i>    |

#### Returns

OPTIX\_ERROR\_INVALID\_VALUE will be returned for traversable handles that are not from acceleration structure builds.

### 8.17.2.6 optixAccelRelocate()

```
OptixResult optixAccelRelocate (
    OptixDeviceContext context,
    CUstream stream,
    const OptixRelocationInfo * info,
    const OptixRelocateInput * relocateInputs,
    size_t numRelocateInputs,
    CUdeviceptr targetAccel,
    size_t targetAccelSizeInBytes,
    OptixTraversableHandle * targetHandle )
```

optixAccelRelocate is called to update the acceleration structure after it has been relocated. Relocation is necessary when the acceleration structure's location in device memory has changed.

optixAccelRelocate does not copy the memory. This function only operates on the relocated memory whose new location is specified by 'targetAccel'. optixAccelRelocate also returns the new OptixTraversableHandle associated with 'targetAccel'. The original memory (source) is not required to be valid, only the [OptixRelocationInfo](#).

Before calling optixAccelRelocate, optixCheckRelocationCompatibility should be called to ensure the copy will be compatible with the destination device context.

The memory pointed to by 'targetAccel' should be allocated with the same size as the source acceleration. Similar to the 'outputBuffer' used in optixAccelBuild, this pointer must be a multiple of OPTIX\_ACCEL\_BUFFER\_BYTE\_ALIGNMENT.

The memory in 'targetAccel' must be allocated as long as the accel is in use.

The instance traversables referenced by an IAS and the micromaps referenced by a triangle GAS may themselves require relocation. 'relocateInputs' and 'numRelocateInputs' should be used to specify the relocated traversables and micromaps. After relocation, the relocated accel will reference these relocated traversables and micromaps instead of their sources. The number of relocate inputs 'numRelocateInputs' must match the number of build inputs 'numBuildInputs' used to build the source accel. Relocation inputs correspond with build inputs used to build the source accel and should appear in the same order (see [optixAccelBuild](#)). 'relocateInputs' and 'numRelocateInputs' may be zero, preserving any references to traversables and micromaps from the source accel.

#### Parameters

|    |                |
|----|----------------|
| in | <i>context</i> |
|----|----------------|

## Parameters

|     |                               |
|-----|-------------------------------|
| in  | <i>stream</i>                 |
| in  | <i>info</i>                   |
| in  | <i>relocateInputs</i>         |
| in  | <i>numRelocateInputs</i>      |
| in  | <i>targetAccel</i>            |
| in  | <i>targetAccelSizeInBytes</i> |
| out | <i>targetHandle</i>           |

## 8.17.2.7 optixBuiltinISModuleGet()

```
OptixResult optixBuiltinISModuleGet (
    OptixDeviceContext context,
    const OptixModuleCompileOptions * moduleCompileOptions,
    const OptixPipelineCompileOptions * pipelineCompileOptions,
    const OptixBuiltinISOptions * builtinISOptions,
    OptixModule * builtinModule )
```

Returns a module containing the intersection program for the built-in primitive type specified by the builtinISOptions. This module must be used as the moduleIS for the [OptixProgramGroupHitgroup](#) in any SBT record for that primitive type. (The entryFunctionNameIS should be null.)

## 8.17.2.8 optixCheckRelocationCompatibility()

```
OptixResult optixCheckRelocationCompatibility (
    OptixDeviceContext context,
    const OptixRelocationInfo * info,
    int * compatible )
```

Checks if an optix data structure built using another OptixDeviceContext (that was used to fill in 'info') is compatible with the OptixDeviceContext specified in the 'context' parameter.

Any device is always compatible with itself.

## Parameters

|     |                   |                                                                                                                                                                                                                                                                       |
|-----|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in  | <i>context</i>    |                                                                                                                                                                                                                                                                       |
| in  | <i>info</i>       |                                                                                                                                                                                                                                                                       |
| out | <i>compatible</i> | <p>If OPTIX_SUCCESS is returned 'compatible' will have the value of either:</p> <ul style="list-style-type: none"> <li>• 0: This context is not compatible with the optix data structure associated with 'info'.</li> <li>• 1: This context is compatible.</li> </ul> |

## 8.17.2.9 optixConvertPointerToTraversableHandle()

```
OptixResult optixConvertPointerToTraversableHandle (
    OptixDeviceContext onDevice,
```

```
CUdeviceptr pointer,
OptixTraversableType traversableType,
OptixTraversableHandle * traversableHandle )
```

## Parameters

|     |                          |                                                                                                                           |
|-----|--------------------------|---------------------------------------------------------------------------------------------------------------------------|
| in  | <i>onDevice</i>          |                                                                                                                           |
| in  | <i>pointer</i>           | pointer to traversable allocated in OptixDeviceContext. This pointer must be a multiple of OPTIX_TRANSFORM_BYTE_ALIGNMENT |
| in  | <i>traversableType</i>   | Type of OptixTraversableHandle to create                                                                                  |
| out | <i>traversableHandle</i> | traversable handle. traversableHandle must be in host memory                                                              |

## 8.17.2.10 optixDenoiserComputeAverageColor()

```
OptixResult optixDenoiserComputeAverageColor (
    OptixDenoiser denoiser,
    CUstream stream,
    const OptixImage2D * inputImage,
    CUdeviceptr outputAverageColor,
    CUdeviceptr scratch,
    size_t scratchSizeInBytes )
```

Compute average logarithmic for each of the first three channels for the given image. When denoising tiles the intensity of the entire image should be computed, i.e. not per tile to get consistent results.

The size of scratch memory required can be queried with [optixDenoiserComputeMemoryResources](#).

data type unsigned char is not supported for 'inputImage', it must be 3 or 4 component half/float.

## Parameters

|     |                           |              |
|-----|---------------------------|--------------|
| in  | <i>denoiser</i>           |              |
| in  | <i>stream</i>             |              |
| in  | <i>inputImage</i>         |              |
| out | <i>outputAverageColor</i> | three floats |
| in  | <i>scratch</i>            |              |
| in  | <i>scratchSizeInBytes</i> |              |

## 8.17.2.11 optixDenoiserComputeIntensity()

```
OptixResult optixDenoiserComputeIntensity (
    OptixDenoiser denoiser,
    CUstream stream,
    const OptixImage2D * inputImage,
    CUdeviceptr outputIntensity,
    CUdeviceptr scratch,
    size_t scratchSizeInBytes )
```

Computes the logarithmic average intensity of the given image. The returned value 'outputIntensity' is multiplied with the RGB values of the input image/tile in `optixDenoiserInvoke` if given in the parameter `OptixDenoiserParams::hdrIntensity` (otherwise 'hdrIntensity' must be a null pointer). This is useful for denoising HDR images which are very dark or bright. When denoising tiles the intensity of the entire image should be computed, i.e. not per tile to get consistent results.

For each RGB pixel in the `inputImage` the intensity is calculated and summed if it is greater than  $1e-8f$ :  $intensity = \log(r * 0.212586f + g * 0.715170f + b * 0.072200f)$ . The function returns  $0.18 / \exp(\text{sum of intensities} / \text{number of summed pixels})$ . More details could be found in the Reinhard tonemapping paper: [http://www.cmap.polytechnique.fr/~peyre/cours/x2005signal/hdr\\_photographic.pdf](http://www.cmap.polytechnique.fr/~peyre/cours/x2005signal/hdr_photographic.pdf)

The size of scratch memory required can be queried with `optixDenoiserComputeMemoryResources`. data type unsigned char is not supported for 'inputImage', it must be 3 or 4 component half/float.

#### Parameters

|     |                           |              |
|-----|---------------------------|--------------|
| in  | <i>denoiser</i>           |              |
| in  | <i>stream</i>             |              |
| in  | <i>inputImage</i>         |              |
| out | <i>outputIntensity</i>    | single float |
| in  | <i>scratch</i>            |              |
| in  | <i>scratchSizeInBytes</i> |              |

#### 8.17.2.12 optixDenoiserComputeMemoryResources()

```
OptixResult optixDenoiserComputeMemoryResources (
    const OptixDenoiser denoiser,
    unsigned int outputWidth,
    unsigned int outputHeight,
    OptixDenoiserSizes * returnSizes )
```

Computes the GPU memory resources required to execute the denoiser.

Memory for state and scratch buffers must be allocated with the sizes in 'returnSizes' and scratch memory passed to `optixDenoiserSetup`, `optixDenoiserInvoke`, `optixDenoiserComputeIntensity` and `optixDenoiserComputeAverageColor`. For tiled denoising an overlap area ('overlapWindowSizeInPixels') must be added to each tile on all sides which increases the amount of memory needed to denoise a tile. In case of tiling use `withOverlapScratchSizeInBytes` for scratch memory size. If only full resolution images are denoised, `withoutOverlapScratchSizeInBytes` can be used which is always smaller than `withOverlapScratchSizeInBytes`.

'outputWidth' and 'outputHeight' is the dimension of the image to be denoised (without overlap in case tiling is being used). 'outputWidth' and 'outputHeight' must be greater than or equal to the dimensions passed to `optixDenoiserSetup`.

#### Parameters

|    |                     |
|----|---------------------|
| in | <i>denoiser</i>     |
| in | <i>outputWidth</i>  |
| in | <i>outputHeight</i> |

## Parameters

|     |                    |
|-----|--------------------|
| out | <i>returnSizes</i> |
|-----|--------------------|

## 8.17.2.13 optixDenoiserCreate()

```
OptixResult optixDenoiserCreate (
    OptixDeviceContext context,
    OptixDenoiserModelKind modelKind,
    const OptixDenoiserOptions * options,
    OptixDenoiser * denoiser )
```

Creates a denoiser object with the given options, using built-in inference models.

'modelKind' selects the model used for inference. Inference for the built-in models can be guided (giving hints to improve image quality) with albedo and normal vector images in the guide layer (see 'optixDenoiserInvoke'). Use of these images must be enabled in 'OptixDenoiserOptions'.

## Parameters

|     |                  |
|-----|------------------|
| in  | <i>context</i>   |
| in  | <i>modelKind</i> |
| in  | <i>options</i>   |
| out | <i>denoiser</i>  |

## 8.17.2.14 optixDenoiserCreateWithUserModel()

```
OptixResult optixDenoiserCreateWithUserModel (
    OptixDeviceContext context,
    const void * userData,
    size_t userDataSizeInBytes,
    OptixDenoiser * denoiser )
```

Creates a denoiser object with the given options, using a provided inference model.

'userData' and 'userDataSizeInBytes' provide a user model for inference. The memory passed in userData will be accessed only during the invocation of this function and can be freed after it returns. The user model must export only one weight set which determines both the model kind and the required set of guide images.

## Parameters

|     |                            |
|-----|----------------------------|
| in  | <i>context</i>             |
| in  | <i>userData</i>            |
| in  | <i>userDataSizeInBytes</i> |
| out | <i>denoiser</i>            |

## 8.17.2.15 optixDenoiserDestroy()

```
OptixResult optixDenoiserDestroy (
```

```
OptixDenoiser denoiser )
```

Destroys the denoiser object and any associated host resources.

#### 8.17.2.16 optixDenoiserInvoke()

```
OptixResult optixDenoiserInvoke (
    OptixDenoiser denoiser,
    CUstream stream,
    const OptixDenoiserParams * params,
    CUdeviceptr denoiserState,
    size_t denoiserStateSizeInBytes,
    const OptixDenoiserGuideLayer * guideLayer,
    const OptixDenoiserLayer * layers,
    unsigned int numLayers,
    unsigned int inputOffsetX,
    unsigned int inputOffsetY,
    CUdeviceptr scratch,
    size_t scratchSizeInBytes )
```

Invokes denoiser on a set of input data and produces at least one output image. State memory must be available during the execution of the denoiser (or until `optixDenoiserSetup` is called with a new state memory pointer). Scratch memory passed is used only for the duration of this function. Scratch and state memory sizes must have a size greater than or equal to the sizes as returned by `optixDenoiserComputeMemoryResources`.

'inputOffsetX' and 'inputOffsetY' are pixel offsets in the 'inputLayers' image specifying the beginning of the image without overlap. When denoising an entire image without tiling there is no overlap and 'inputOffsetX' and 'inputOffsetY' must be zero. When denoising a tile which is adjacent to one of the four sides of the entire image the corresponding offsets must also be zero since there is no overlap at the side adjacent to the image border.

'guideLayer' provides additional information to the denoiser. When providing albedo and normal vector guide images, the corresponding fields in the '`OptixDenoiserOptions`' must be enabled, see `optixDenoiserCreate`. 'guideLayer' must not be null. If a guide image in '`OptixDenoiserOptions`' is not enabled, the corresponding image in '`OptixDenoiserGuideLayer`' is ignored.

If `OPTIX_DENOISER_MODEL_KIND_TEMPORAL` or `OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV` is selected, a 2d flow image must be given in '`OptixDenoiserGuideLayer`'. It describes for each pixel the flow from the previous to the current frame (a 2d vector in pixel space). The denoised beauty/AOV of the previous frame must be given in 'previousOutput'. If this image is not available in the first frame of a sequence, the noisy beauty/AOV from the first frame and zero flow vectors could be given as a substitute. For non-temporal model kinds the flow image in '`OptixDenoiserGuideLayer`' is ignored. 'previousOutput' and 'output' may refer to the same buffer if tiling is not used, i.e. 'previousOutput' is first read by this function and later overwritten with the denoised result. 'output' can be passed as 'previousOutput' to the next frame. In other model kinds (not temporal) 'previousOutput' is ignored.

The beauty layer must be given as the first entry in 'layers'. In AOV type model kinds (`OPTIX_DENOISER_MODEL_KIND_AOV` or in user defined models implementing kernel-prediction) additional layers for the AOV images can be given. In each layer the noisy input image is given in 'input', the denoised output is written into the 'output' image. input and output images may refer to the same buffer, with the restriction that the pixel formats must be identical for input and output when

the blend mode is selected (see [OptixDenoiserParams](#)).

If `OPTIX_DENOISER_MODEL_KIND_TEMPORAL` or `OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV` is selected, the denoised image from the previous frame must be given in 'previousOutput' in the layer. 'previousOutput' and 'output' may refer to the same buffer if tiling is not used, i.e. 'previousOutput' is first read by this function and later overwritten with the denoised result. 'output' can be passed as 'previousOutput' to the next frame. In addition, 'previousOutputInternalGuideLayer' and 'outputInternalGuideLayer' must both be allocated regardless of tiling mode. The pixel format must be `OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER` and the dimension must be identical to the other input layers. In the first frame memory in 'previousOutputInternalGuideLayer' must either contain valid data from previous denoiser runs or set to zero. In other model kinds (not temporal) 'previousOutput' and the internal guide layers are ignored.

If `OPTIX_DENOISER_MODEL_KIND_TEMPORAL` or `OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV` is selected, the normal vector guide image must be given as 3d vectors in camera space. In the other models only the x and y channels are used and other channels are ignored.

#### Parameters

|    |                                 |
|----|---------------------------------|
| in | <i>denoiser</i>                 |
| in | <i>stream</i>                   |
| in | <i>params</i>                   |
| in | <i>denoiserState</i>            |
| in | <i>denoiserStateSizeInBytes</i> |
| in | <i>guideLayer</i>               |
| in | <i>layers</i>                   |
| in | <i>numLayers</i>                |
| in | <i>inputOffsetX</i>             |
| in | <i>inputOffsetY</i>             |
| in | <i>scratch</i>                  |
| in | <i>scratchSizeInBytes</i>       |

#### 8.17.2.17 optixDenoiserSetup()

```
OptixResult optixDenoiserSetup (
    OptixDenoiser denoiser,
    CUstream stream,
    unsigned int inputWidth,
    unsigned int inputHeight,
    CUdeviceptr denoiserState,
    size_t denoiserStateSizeInBytes,
    CUdeviceptr scratch,
    size_t scratchSizeInBytes )
```

Initializes the state required by the denoiser.

'inputWidth' and 'inputHeight' must include overlap on both sides of the image if tiling is being used. The overlap is returned by [optixDenoiserComputeMemoryResources](#). For subsequent calls to [optixDenoiserInvoke](#) 'inputWidth' and 'inputHeight' are the maximum dimensions of the input layers.

Dimensions of the input layers passed to `optixDenoiserInvoke` may be different in each invocation however they always must be smaller than 'inputWidth' and 'inputHeight' passed to `optixDenoiserSetup`.

#### Parameters

|    |                                 |
|----|---------------------------------|
| in | <i>denoiser</i>                 |
| in | <i>stream</i>                   |
| in | <i>inputWidth</i>               |
| in | <i>inputHeight</i>              |
| in | <i>denoiserState</i>            |
| in | <i>denoiserStateSizeInBytes</i> |
| in | <i>scratch</i>                  |
| in | <i>scratchSizeInBytes</i>       |

#### 8.17.2.18 optixDeviceContextCreate()

```
OptixResult optixDeviceContextCreate (
    CUcontext fromContext,
    const OptixDeviceContextOptions * options,
    OptixDeviceContext * context )
```

Create a device context associated with the CUDA context specified with 'fromContext'.

If zero is specified for 'fromContext', OptiX will use the current CUDA context. The CUDA context should be initialized before calling `optixDeviceContextCreate`.

#### Parameters

|     |                    |
|-----|--------------------|
| in  | <i>fromContext</i> |
| in  | <i>options</i>     |
| out | <i>context</i>     |

#### Returns

- `OPTIX_ERROR_CUDA_NOT_INITIALIZED` If using zero for 'fromContext' and CUDA has not been initialized yet on the calling thread.
- `OPTIX_ERROR_CUDA_ERROR` CUDA operation failed.
- `OPTIX_ERROR_HOST_OUT_OF_MEMORY` Heap allocation failed.
- `OPTIX_ERROR_INTERNAL_ERROR` Internal error

#### 8.17.2.19 optixDeviceContextDestroy()

```
OptixResult optixDeviceContextDestroy (
    OptixDeviceContext context )
```

Destroys all CPU and GPU state associated with the device.

It will attempt to block on CUDA streams that have launch work outstanding.

Any API objects, such as `OptixModule` and `OptixPipeline`, not already destroyed will be destroyed.



Thread safety: A device context must not be destroyed while it is still in use by concurrent API calls in other threads.

#### 8.17.2.20 optixDeviceContextGetCacheDatabaseSizes()

```
OptixResult optixDeviceContextGetCacheDatabaseSizes (
    OptixDeviceContext context,
    size_t * lowWaterMark,
    size_t * highWaterMark )
```

Returns the low and high water marks for disk cache garbage collection. If the cache has been disabled by setting the environment variable OPTIX\_CACHE\_MAXSIZE=0, this function will return 0 for the low and high water marks.

Parameters

|     |                      |                     |
|-----|----------------------|---------------------|
| in  | <i>context</i>       | the device context  |
| out | <i>lowWaterMark</i>  | the low water mark  |
| out | <i>highWaterMark</i> | the high water mark |

#### 8.17.2.21 optixDeviceContextGetCacheEnabled()

```
OptixResult optixDeviceContextGetCacheEnabled (
    OptixDeviceContext context,
    int * enabled )
```

Indicates whether the disk cache is enabled or disabled.

Parameters

|     |                |                             |
|-----|----------------|-----------------------------|
| in  | <i>context</i> | the device context          |
| out | <i>enabled</i> | 1 if enabled, 0 if disabled |

#### 8.17.2.22 optixDeviceContextGetCacheLocation()

```
OptixResult optixDeviceContextGetCacheLocation (
    OptixDeviceContext context,
    char * location,
    size_t locationSize )
```

Returns the location of the disk cache. If the cache has been disabled by setting the environment variable OPTIX\_CACHE\_MAXSIZE=0, this function will return an empty string.

Parameters

|     |                     |                                                              |
|-----|---------------------|--------------------------------------------------------------|
| in  | <i>context</i>      | the device context                                           |
| out | <i>location</i>     | directory of disk cache, null terminated if locationSize > 0 |
| in  | <i>locationSize</i> | locationSize                                                 |

### 8.17.2.23 optixDeviceContextGetProperty()

```
OptixResult optixDeviceContextGetProperty (
    OptixDeviceContext context,
    OptixDeviceProperty property,
    void * value,
    size_t sizeInBytes )
```

Query properties of a device context.

Parameters

|     |                    |                                              |
|-----|--------------------|----------------------------------------------|
| in  | <i>context</i>     | the device context to query the property for |
| in  | <i>property</i>    | the property to query                        |
| out | <i>value</i>       | pointer to the returned                      |
| in  | <i>sizeInBytes</i> | size of output                               |

### 8.17.2.24 optixDeviceContextSetCacheDatabaseSizes()

```
OptixResult optixDeviceContextSetCacheDatabaseSizes (
    OptixDeviceContext context,
    size_t lowWaterMark,
    size_t highWaterMark )
```

Sets the low and high water marks for disk cache garbage collection.

Garbage collection is triggered when a new entry is written to the cache and the current cache data size plus the size of the cache entry that is about to be inserted exceeds the high water mark. Garbage collection proceeds until the size reaches the low water mark. Garbage collection will always free enough space to insert the new entry without exceeding the low water mark. Setting either limit to zero will disable garbage collection. An error will be returned if both limits are non-zero and the high water mark is smaller than the low water mark.

Note that garbage collection is performed only on writes to the disk cache. No garbage collection is triggered on disk cache initialization or immediately when calling this function, but on subsequent inserting of data into the database.

If the size of a compiled module exceeds the value configured for the high water mark and garbage collection is enabled, the module will not be added to the cache and a warning will be added to the log.

The high water mark can be overridden with the environment variable `OPTIX_CACHE_MAXSIZE`. The environment variable takes precedence over the function parameters. The low water mark will be set to half the value of `OPTIX_CACHE_MAXSIZE`. Setting `OPTIX_CACHE_MAXSIZE` to 0 will disable the disk cache, but will not alter the contents of the cache. Negative and non-integer values will be ignored.

Parameters

|    |                      |                     |
|----|----------------------|---------------------|
| in | <i>context</i>       | the device context  |
| in | <i>lowWaterMark</i>  | the low water mark  |
| in | <i>highWaterMark</i> | the high water mark |

### 8.17.2.25 optixDeviceContextSetCacheEnabled()

```
OptixResult optixDeviceContextSetCacheEnabled (
    OptixDeviceContext context,
    int enabled )
```

Enables or disables the disk cache.

If caching was previously disabled, enabling it will attempt to initialize the disk cache database using the currently configured cache location. An error will be returned if initialization fails.

Note that no in-memory cache is used, so no caching behavior will be observed if the disk cache is disabled.

The cache can be disabled by setting the environment variable OPTIX\_CACHE\_MAXSIZE=0. The environment variable takes precedence over this setting. See [optixDeviceContextSetCacheDatabaseSizes](#) for additional information.

Note that the disk cache can be disabled by the environment variable, but it cannot be enabled via the environment if it is disabled via the API.

#### Parameters

|    |                |                            |
|----|----------------|----------------------------|
| in | <i>context</i> | the device context         |
| in | <i>enabled</i> | 1 to enabled, 0 to disable |

### 8.17.2.26 optixDeviceContextSetCacheLocation()

```
OptixResult optixDeviceContextSetCacheLocation (
    OptixDeviceContext context,
    const char * location )
```

Sets the location of the disk cache.

The location is specified by a directory. This directory should not be used for other purposes and will be created if it does not exist. An error will be returned if it is not possible to create the disk cache at the specified location for any reason (e.g., the path is invalid or the directory is not writable). Caching will be disabled if the disk cache cannot be initialized in the new location. If caching is disabled, no error will be returned until caching is enabled. If the disk cache is located on a network file share, behavior is undefined.

The location of the disk cache can be overridden with the environment variable OPTIX\_CACHE\_PATH. The environment variable takes precedence over this setting.

The default location depends on the operating system:

- Windows: LOCALAPPDATA%\NVIDIA\OptixCache
- Linux: /var/tmp/OptixCache\_<username> (or /tmp/OptixCache\_<username> if the first choice is not usable), the underscore and username suffix are omitted if the username cannot be obtained
- MacOS X: /Library/Application Support/NVIDIA/OptixCache

#### Parameters

|    |                 |                         |
|----|-----------------|-------------------------|
| in | <i>context</i>  | the device context      |
| in | <i>location</i> | directory of disk cache |

### 8.17.2.27 optixDeviceContextSetLogCallback()

```
OptixResult optixDeviceContextSetLogCallback (
    OptixDeviceContext context,
    OptixLogCallback callbackFunction,
    void * callbackData,
    unsigned int callbackLevel )
```

Sets the current log callback method.

See [OptixLogCallback](#) for more details.

Thread safety: It is guaranteed that the callback itself (callbackFunction and callbackData) are updated atomically. It is not guaranteed that the callback itself (callbackFunction and callbackData) and the callbackLevel are updated atomically. It is unspecified when concurrent API calls using the same context start to make use of the new callback method.

Parameters

|    |                         |                                                               |
|----|-------------------------|---------------------------------------------------------------|
| in | <i>context</i>          | the device context                                            |
| in | <i>callbackFunction</i> | the callback function to call                                 |
| in | <i>callbackData</i>     | pointer to data passed to callback function while invoking it |
| in | <i>callbackLevel</i>    | callback level                                                |

### 8.17.2.28 optixDisplacementMicromapArrayBuild()

```
OptixResult optixDisplacementMicromapArrayBuild (
    OptixDeviceContext context,
    CUstream stream,
    const OptixDisplacementMicromapArrayBuildInput * buildInput,
    const OptixMicromapBuffers * buffers )
```

FIXME Construct an array of Displacement Micromap (DMMs).

Each triangle within a DMM GAS geometry references one DMM that specifies how to subdivide it into micro-triangles. A DMM gives a subdivision resolution into  $4^N$  micro-triangles, and displacement values for each of the vertices in the subdivided mesh. The values are combined with e.g. normal vectors, scale and bias given as AS build inputs, to get the final geometry. A DMM is encoded in one or more compressed blocks, each block having displacement values for a subtriangle of 64..1024 micro-triangles.

Parameters

|    |                   |                                                   |
|----|-------------------|---------------------------------------------------|
| in | <i>context</i>    |                                                   |
| in | <i>stream</i>     |                                                   |
| in | <i>buildInput</i> | a single build input object referencing many DMMs |
| in | <i>buffers</i>    | the buffers used for build                        |

### 8.17.2.29 optixDisplacementMicromapArrayComputeMemoryUsage()

```
OptixResult optixDisplacementMicromapArrayComputeMemoryUsage (
```

```

    OptixDeviceContext context,
    const OptixDisplacementMicromapArrayBuildInput * buildInput,
    OptixMicromapBufferSizes * bufferSizes )

```

Determine the amount of memory necessary for a Displacement Micromap Array build.

Parameters

|     |                    |
|-----|--------------------|
| in  | <i>context</i>     |
| in  | <i>buildInput</i>  |
| out | <i>bufferSizes</i> |

### 8.17.2.30 optixGetErrorName()

```

const char * optixGetErrorName (
    OptixResult result )

```

Returns a string containing the name of an error code in the enum.

Output is a string representation of the enum. For example "OPTIX\_SUCCESS" for OPTIX\_SUCCESS and "OPTIX\_ERROR\_INVALID\_VALUE" for OPTIX\_ERROR\_INVALID\_VALUE.

If the error code is not recognized, "Unrecognized OptixResult code" is returned.

Parameters

|    |               |                                              |
|----|---------------|----------------------------------------------|
| in | <i>result</i> | OptixResult enum to generate string name for |
|----|---------------|----------------------------------------------|

See also [optixGetErrorString](#)

### 8.17.2.31 optixGetErrorString()

```

const char * optixGetErrorString (
    OptixResult result )

```

Returns the description string for an error code.

Output is a string description of the enum. For example "Success" for OPTIX\_SUCCESS and "Invalid value" for OPTIX\_ERROR\_INVALID\_VALUE.

If the error code is not recognized, "Unrecognized OptixResult code" is returned.

Parameters

|    |               |                                                     |
|----|---------------|-----------------------------------------------------|
| in | <i>result</i> | OptixResult enum to generate string description for |
|----|---------------|-----------------------------------------------------|

See also [optixGetErrorName](#)

### 8.17.2.32 optixLaunch()

```

OptixResult optixLaunch (
    OptixPipeline pipeline,
    CUstream stream,
    CUdeviceptr pipelineParams,

```

```

size_t pipelineParamsSize,
const OptixShaderBindingTable * sbt,
unsigned int width,
unsigned int height,
unsigned int depth )

```

Where the magic happens.

The stream and pipeline must belong to the same device context. Multiple launches may be issues in parallel from multiple threads to different streams.

pipelineParamsSize number of bytes are copied from the device memory pointed to by pipelineParams before launch. It is an error if pipelineParamsSize is greater than the size of the variable declared in modules and identified by `OptixPipelineCompileOptions::pipelineLaunchParamsVariableName`. If the launch params variable was optimized out or not found in the modules linked to the pipeline then the pipelineParams and pipelineParamsSize parameters are ignored.

sbt points to the shader binding table, which defines shader groupings and their resources. See the SBT spec.

#### Parameters

|    |                           |                               |
|----|---------------------------|-------------------------------|
| in | <i>pipeline</i>           |                               |
| in | <i>stream</i>             |                               |
| in | <i>pipelineParams</i>     |                               |
| in | <i>pipelineParamsSize</i> |                               |
| in | <i>sbt</i>                |                               |
| in | <i>width</i>              | number of elements to compute |
| in | <i>height</i>             | number of elements to compute |
| in | <i>depth</i>              | number of elements to compute |

Thread safety: In the current implementation concurrent launches to the same pipeline are not supported. Concurrent launches require separate OptixPipeline objects.

#### 8.17.2.33 optixModuleCreate()

```

OptixResult optixModuleCreate (
    OptixDeviceContext context,
    const OptixModuleCompileOptions * moduleCompileOptions,
    const OptixPipelineCompileOptions * pipelineCompileOptions,
    const char * input,
    size_t inputSize,
    char * logString,
    size_t * logStringSize,
    OptixModule * module )

```

Compiling programs into a module. These programs can be passed in as either PTX or OptiX-IR.

See the Programming Guide for details, as well as how to generate these encodings from CUDA sources.

logString is an optional buffer that contains compiler feedback and errors. This information is also

passed to the context logger (if enabled), however it may be difficult to correlate output to the logger to specific API invocations when using multiple threads. The output to logString will only contain feedback for this specific invocation of this API call.

logStringSize as input should be a pointer to the number of bytes backing logString. Upon return it contains the length of the log message (including the null terminator) which may be greater than the input value. In this case, the log message will be truncated to fit into logString.

If logString or logStringSize are NULL, no output is written to logString. If logStringSize points to a value that is zero, no output is written. This does not affect output to the context logger if enabled.

#### Parameters

|         |                               |                                                                                                                      |
|---------|-------------------------------|----------------------------------------------------------------------------------------------------------------------|
| in      | <i>context</i>                |                                                                                                                      |
| in      | <i>moduleCompileOptions</i>   |                                                                                                                      |
| in      | <i>pipelineCompileOptions</i> | All modules in a pipeline need to use the same values for the pipeline compile options.                              |
| in      | <i>input</i>                  | Pointer to the input code.                                                                                           |
| in      | <i>inputSize</i>              | Parsing proceeds up to inputSize characters. Or, when reading PTX input, the first NUL byte, whichever occurs first. |
| out     | <i>logString</i>              | Information will be written to this string. If logStringSize > 0 logString will be null terminated.                  |
| in, out | <i>logStringSize</i>          |                                                                                                                      |
| out     | <i>module</i>                 |                                                                                                                      |

#### Returns

OPTIX\_ERROR\_INVALID\_VALUE - context is 0, moduleCompileOptions is 0, pipelineCompileOptions is 0, input is 0, module is 0.

### 8.17.2.34 optixModuleCreateWithTasks()

```
OptixResult optixModuleCreateWithTasks (
    OptixDeviceContext context,
    const OptixModuleCompileOptions * moduleCompileOptions,
    const OptixPipelineCompileOptions * pipelineCompileOptions,
    const char * input,
    size_t inputSize,
    char * logString,
    size_t * logStringSize,
    OptixModule * module,
    OptixTask * firstTask )
```

This function is designed to do just enough work to create the OptixTask return parameter and is expected to be fast enough run without needing parallel execution. A single thread could generate all the OptixTask objects for further processing in a work pool.

Options are similar to [optixModuleCreate\(\)](#), aside from the return parameter, firstTask.

The memory used to hold the input should be live until all tasks are finished.

It is illegal to call `optixModuleDestroy()` if any `OptixTask` objects are currently being executed. In that case `OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE` will be returned.

If an invocation of `optixTaskExecute` fails, the `OptixModule` will be marked as `OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE` if there are outstanding tasks or `OPTIX_MODULE_COMPILE_STATE_FAILURE` if there are no outstanding tasks. Subsequent calls to `optixTaskExecute()` may execute additional work to collect compilation errors generated from the input. Currently executing tasks will not necessarily be terminated immediately but at the next opportunity. Logging will continue to be directed to the logger installed with the `OptixDeviceContext`. If `logString` is provided to `optixModuleCreateWithTasks()`, it will contain all the compiler feedback from all executed tasks. The lifetime of the memory pointed to by `logString` should extend from calling `optixModuleCreateWithTasks()` to when the compilation state is either `OPTIX_MODULE_COMPILE_STATE_FAILURE` or `OPTIX_MODULE_COMPILE_STATE_COMPLETED`. OptiX will not write to the `logString` outside of execution of `optixModuleCreateWithTasks()` or `optixTaskExecute()`. If the compilation state is `OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE` and no further execution of `optixTaskExecute()` is performed the `logString` may be reclaimed by the application before calling `optixModuleDestroy()`. The contents of `logString` will contain output from currently completed tasks. All `OptixTask` objects associated with a given `OptixModule` will be cleaned up when `optixModuleDestroy()` is called regardless of whether the compilation was successful or not. If the compilation state is `OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE`, any unstarted `OptixTask` objects do not need to be executed though there is no harm doing so.

See also `optixModuleCreate`

### 8.17.2.35 optixModuleDestroy()

```
OptixResult optixModuleDestroy (
    OptixModule module )
```

Call for `OptixModule` objects created with `optixModuleCreate` and `optixModuleDeserialize`.

Modules must not be destroyed while they are still used by any program group.

Thread safety: A module must not be destroyed while it is still in use by concurrent API calls in other threads.

### 8.17.2.36 optixModuleGetCompilationState()

```
OptixResult optixModuleGetCompilationState (
    OptixModule module,
    OptixModuleCompileState * state )
```

When creating a module with tasks, the current state of the module can be queried using this function.

Thread safety: Safe to call from any thread until `optixModuleDestroy` is called.

See also `optixModuleCreateWithTasks`

### 8.17.2.37 optixOpacityMicromapArrayBuild()

```
OptixResult optixOpacityMicromapArrayBuild (
    OptixDeviceContext context,
    CUstream stream,
    const OptixOpacityMicromapArrayBuildInput * buildInput,
    const OptixMicromapBuffers * buffers )
```

Construct an array of Opacity Micromaps.



Each triangle within an instance/GAS may reference one opacity micromap to give finer control over alpha behavior. A opacity micromap consists of a set of  $4^N$  micro-triangles in a triangular uniform barycentric grid. Multiple opacity micromaps are collected (built) into a opacity micromap array with this function. Each geometry in a GAS may bind a single opacity micromap array and can use opacity micromaps from that array only.

Each micro-triangle within a opacity micromap can be in one of four states: Transparent, Opaque, Unknown-Transparent or Unknown-Opaque. During traversal, if a triangle with a opacity micromap attached is intersected, the opacity micromap is queried to categorize the hit as either opaque, unknown (alpha) or a miss. Geometry, ray or instance flags that modify the alpha/opaque behavior are applied *after* this opacity micromap query.

The opacity micromap query may operate in 2-state mode (alpha testing) or 4-state mode (AHS culling), depending on the opacity micromap type and ray/instance flags. When operating in 2-state mode, alpha hits will not be reported, and transparent and opaque hits must be accurate.

#### Parameters

|    |                   |                                                                |
|----|-------------------|----------------------------------------------------------------|
| in | <i>context</i>    |                                                                |
| in | <i>stream</i>     |                                                                |
| in | <i>buildInput</i> | a single build input object referencing many opacity micromaps |
| in | <i>buffers</i>    | the buffers used for build                                     |

### 8.17.2.38 optixOpacityMicromapArrayComputeMemoryUsage()

```
OptixResult optixOpacityMicromapArrayComputeMemoryUsage (
    OptixDeviceContext context,
    const OptixOpacityMicromapArrayBuildInput * buildInput,
    OptixMicromapBufferSizes * bufferSizes )
```

Determine the amount of memory necessary for a Opacity Micromap Array build.

#### Parameters

|     |                    |
|-----|--------------------|
| in  | <i>context</i>     |
| in  | <i>buildInput</i>  |
| out | <i>bufferSizes</i> |

### 8.17.2.39 optixOpacityMicromapArrayGetRelocationInfo()

```
OptixResult optixOpacityMicromapArrayGetRelocationInfo (
    OptixDeviceContext context,
    CUdeviceptr opacityMicromapArray,
    OptixRelocationInfo * info )
```

Obtain relocation information, stored in [OptixRelocationInfo](#), for a given context and opacity micromap array.

The relocation information can be passed to [optixCheckRelocationCompatibility](#) to determine if a opacity micromap array, referenced by buffers, can be relocated to a different device's memory space (see [optixCheckRelocationCompatibility](#)).

When used with `optixOpacityMicromapArrayRelocate`, it provides data necessary for doing the relocation.

If the opacity micromap array data associated with 'opacityMicromapArray' is copied multiple times, the same [OptixRelocationInfo](#) can also be used on all copies.

#### Parameters

|     |                             |
|-----|-----------------------------|
| in  | <i>context</i>              |
| in  | <i>opacityMicromapArray</i> |
| out | <i>info</i>                 |

### 8.17.2.40 optixOpacityMicromapArrayRelocate()

```
OptixResult optixOpacityMicromapArrayRelocate (
    OptixDeviceContext context,
    CUstream stream,
    const OptixRelocationInfo * info,
    CUdeviceptr targetOpacityMicromapArray,
    size_t targetOpacityMicromapArraySizeInBytes )
```

`optixOpacityMicromapArrayRelocate` is called to update the opacity micromap array after it has been relocated. Relocation is necessary when the opacity micromap array's location in device memory has changed. `optixOpacityMicromapArrayRelocate` does not copy the memory. This function only operates on the relocated memory whose new location is specified by 'targetOpacityMicromapArray'. The original memory (source) is not required to be valid, only the [OptixRelocationInfo](#).

Before calling `optixOpacityMicromapArrayRelocate`, `optixCheckRelocationCompatibility` should be called to ensure the copy will be compatible with the destination device context.

The memory pointed to by 'targetOpacityMicromapArray' should be allocated with the same size as the source opacity micromap array. Similar to the '[OptixMicromapBuffers::output](#)' used in `optixOpacityMicromapArrayBuild`, this pointer must be a multiple of `OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT`.

The memory in 'targetOpacityMicromapArray' must be allocated as long as the opacity micromap array is in use.

Note that any Acceleration Structures build using the original memory (source) as input will still be associated with this original memory. To associate an existing (possibly relocated) Acceleration Structures with the relocated opacity micromap array, use `optixAccelBuild` to update the existing Acceleration Structures (See `OPTIX_BUILD_OPERATION_UPDATE`)

#### Parameters

|    |                                              |
|----|----------------------------------------------|
| in | <i>context</i>                               |
| in | <i>stream</i>                                |
| in | <i>info</i>                                  |
| in | <i>targetOpacityMicromapArray</i>            |
| in | <i>targetOpacityMicromapArraySizeInBytes</i> |

## 8.17.2.41 optixPipelineCreate()

```

OptixResult optixPipelineCreate (
    OptixDeviceContext context,
    const OptixPipelineCompileOptions * pipelineCompileOptions,
    const OptixPipelineLinkOptions * pipelineLinkOptions,
    const OptixProgramGroup * programGroups,
    unsigned int numProgramGroups,
    char * logString,
    size_t * logStringSize,
    OptixPipeline * pipeline )

```

logString is an optional buffer that contains compiler feedback and errors. This information is also passed to the context logger (if enabled), however it may be difficult to correlate output to the logger to specific API invocations when using multiple threads. The output to logString will only contain feedback for this specific invocation of this API call.

logStringSize as input should be a pointer to the number of bytes backing logString. Upon return it contains the length of the log message (including the null terminator) which may be greater than the input value. In this case, the log message will be truncated to fit into logString.

If logString or logStringSize are NULL, no output is written to logString. If logStringSize points to a value that is zero, no output is written. This does not affect output to the context logger if enabled.

## Parameters

|         |                               |                                                                                                     |
|---------|-------------------------------|-----------------------------------------------------------------------------------------------------|
| in      | <i>context</i>                |                                                                                                     |
| in      | <i>pipelineCompileOptions</i> |                                                                                                     |
| in      | <i>pipelineLinkOptions</i>    |                                                                                                     |
| in      | <i>programGroups</i>          | array of ProgramGroup objects                                                                       |
| in      | <i>numProgramGroups</i>       | number of ProgramGroup objects                                                                      |
| out     | <i>logString</i>              | Information will be written to this string. If logStringSize > 0 logString will be null terminated. |
| in, out | <i>logStringSize</i>          |                                                                                                     |
| out     | <i>pipeline</i>               |                                                                                                     |

## 8.17.2.42 optixPipelineDestroy()

```

OptixResult optixPipelineDestroy (
    OptixPipeline pipeline )

```

Thread safety: A pipeline must not be destroyed while it is still in use by concurrent API calls in other threads.

## 8.17.2.43 optixPipelineSetStackSize()

```

OptixResult optixPipelineSetStackSize (
    OptixPipeline pipeline,
    unsigned int directCallableStackSizeFromTraversal,
    unsigned int directCallableStackSizeFromState,

```

```

    unsigned int continuationStackSize,
    unsigned int maxTraversableGraphDepth )

```

Sets the stack sizes for a pipeline.

Users are encouraged to see the programming guide and the implementations of the helper functions to understand how to construct the stack sizes based on their particular needs.

If this method is not used, an internal default implementation is used. The default implementation is correct (but not necessarily optimal) as long as the maximum depth of call trees of CC and DC programs is at most 2 and no motion transforms are used.

The maxTraversableGraphDepth responds to the maximal number of traversables visited when calling trace. Every acceleration structure and motion transform count as one level of traversal. E.g., for a simple IAS (instance acceleration structure) -> GAS (geometry acceleration structure) traversal graph, the maxTraversableGraphDepth is two. For IAS -> MT (motion transform) -> GAS, the maxTraversableGraphDepth is three. Note that it does not matter whether a IAS or GAS has motion or not, it always counts as one. Launching optix with exceptions turned on (see [OPTIX\\_EXCEPTION\\_FLAG\\_TRACE\\_DEPTH](#)) will throw an exception if the specified maxTraversableGraphDepth is too small.

#### Parameters

|    |                                             |                                                                                    |
|----|---------------------------------------------|------------------------------------------------------------------------------------|
| in | <i>pipeline</i>                             | The pipeline to configure the stack size for.                                      |
| in | <i>directCallableStackSizeFromTraversal</i> | The direct stack size requirement for direct callables invoked from IS or AH.      |
| in | <i>directCallableStackSizeFromState</i>     | The direct stack size requirement for direct callables invoked from RG, MS, or CH. |
| in | <i>continuationStackSize</i>                | The continuation stack requirement.                                                |
| in | <i>maxTraversableGraphDepth</i>             | The maximum depth of a traversable graph passed to trace.                          |

#### 8.17.2.44 optixProgramGroupCreate()

```

OptixResult optixProgramGroupCreate (
    OptixDeviceContext context,
    const OptixProgramGroupDesc * programDescriptions,
    unsigned int numProgramGroups,
    const OptixProgramGroupOptions * options,
    char * logString,
    size_t * logStringSize,
    OptixProgramGroup * programGroups )

```

logString is an optional buffer that contains compiler feedback and errors. This information is also passed to the context logger (if enabled), however it may be difficult to correlate output to the logger to specific API invocations when using multiple threads. The output to logString will only contain feedback for this specific invocation of this API call.

logStringSize as input should be a pointer to the number of bytes backing logString. Upon return it contains the length of the log message (including the null terminator) which may be greater than the input value. In this case, the log message will be truncated to fit into logString.

If logString or logStringSize are NULL, no output is written to logString. If logStringSize points to a value that is zero, no output is written. This does not affect output to the context logger if enabled.

Creates numProgramGroups OptiXProgramGroup objects from the specified [OptixProgramGroupDesc](#) array. The size of the arrays must match.

#### Parameters

|         |                            |                                                                                                     |
|---------|----------------------------|-----------------------------------------------------------------------------------------------------|
| in      | <i>context</i>             |                                                                                                     |
| in      | <i>programDescriptions</i> | $N * \text{OptixProgramGroupDesc}$                                                                  |
| in      | <i>numProgramGroups</i>    | N                                                                                                   |
| in      | <i>options</i>             |                                                                                                     |
| out     | <i>logString</i>           | Information will be written to this string. If logStringSize > 0 logString will be null terminated. |
| in, out | <i>logStringSize</i>       |                                                                                                     |
| out     | <i>programGroups</i>       |                                                                                                     |

#### 8.17.2.45 optixProgramGroupDestroy()

```
OptixResult optixProgramGroupDestroy (
    OptixProgramGroup programGroup )
```

Thread safety: A program group must not be destroyed while it is still in use by concurrent API calls in other threads.

#### 8.17.2.46 optixProgramGroupGetStackSize()

```
OptixResult optixProgramGroupGetStackSize (
    OptixProgramGroup programGroup,
    OptixStackSizes * stackSizes,
    OptixPipeline pipeline )
```

Returns the stack sizes for the given program group. When programs in this programGroup are relying on external functions, the corresponding stack sizes can only be correctly retrieved when all functions are known after linking, i.e. when a pipeline has been created. When pipeline is set to NULL, the stack size will be calculated excluding external functions. In this case a warning will be issued if external functions are referenced by the OptixModule.

#### Parameters

|     |                     |                                                                      |
|-----|---------------------|----------------------------------------------------------------------|
| in  | <i>programGroup</i> | the program group                                                    |
| out | <i>stackSizes</i>   | the corresponding stack sizes                                        |
| in  | <i>pipeline</i>     | considering the program group within the given pipeline, can be NULL |

#### 8.17.2.47 optixSbtRecordPackHeader()

```
OptixResult optixSbtRecordPackHeader (
    OptixProgramGroup programGroup,
    void * sbtRecordHeaderHostPointer )
```

## Parameters

|     |                                   |                                             |
|-----|-----------------------------------|---------------------------------------------|
| in  | <i>programGroup</i>               | the program group containing the program(s) |
| out | <i>sbtRecordHeaderHostPointer</i> | the result sbt record header                |

## 8.17.2.48 optixTaskExecute()

```
OptixResult optixTaskExecute (
    OptixTask task,
    OptixTask * additionalTasks,
    unsigned int maxNumAdditionalTasks,
    unsigned int * numAdditionalTasksCreated )
```

Each OptixTask should be executed with [optixTaskExecute\(\)](#). If additional parallel work is found, new OptixTask objects will be returned in additionalTasks along with the number of additional tasks in numAdditionalTasksCreated. The parameter additionalTasks should point to a user allocated array of minimum size maxNumAdditionalTasks. OptiX can generate upto maxNumAdditionalTasks additional tasks.

Each task can be executed in parallel and in any order.

Thread safety: Safe to call from any thread until [optixModuleDestroy\(\)](#) is called for any associated task.

See also [optixModuleCreateWithTasks](#)

## Parameters

|     |                                  |                                                                               |
|-----|----------------------------------|-------------------------------------------------------------------------------|
| in  | <i>task</i>                      | the OptixTask to execute                                                      |
| in  | <i>additionalTasks</i>           | pointer to array of OptixTask objects to be filled in                         |
| in  | <i>maxNumAdditionalTasks</i>     | maximum number of additional OptixTask objects                                |
| out | <i>numAdditionalTasksCreated</i> | number of OptixTask objects created by OptiX and written into additionalTasks |

## 8.18 optix\_host.h

[Go to the documentation of this file.](#)

```
1 /*
2  * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
3  *
4  * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5  * rights in and to this software, related documentation and any modifications thereto.
6  * Any use, reproduction, disclosure or distribution of this software and related
7  * documentation without an express license agreement from NVIDIA Corporation is strictly
8  * prohibited.
9  *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE. IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
```

```

27
28 #ifndef OPTIX_OPTIX_HOST_H
29 #define OPTIX_OPTIX_HOST_H
30
31 #include "optix_types.h"
32 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
33 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver types must be defined through other
34 // means before including optix headers.
35 #include <cuda.h>
36 #endif
37
38 #ifdef NV_MODULE_OPTIX
39 // This is a mechanism to include <g_nvconfig.h> in driver builds only and translate any nvconfig macro to
40 // a custom OPTIX-specific macro, that can also be used in SDK builds/installs
41 #include <exp/misc/optix_nvconfig_translate.h> // includes <g_nvconfig.h>
42 #endif // NV_MODULE_OPTIX
43
44 #ifdef __cplusplus
45 extern "C" {
46 #endif
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65 const char* optixGetErrorName(OptixResult result);
66
67 const char* optixGetErrorString(OptixResult result);
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102 OptixResult optixDeviceContextCreate(CUcontext fromContext, const OptixDeviceContextOptions* options,
103 OptixDeviceContext* context);
104
105
106
107
108
109
110
111
112 OptixResult optixDeviceContextDestroy(OptixDeviceContext context);
113
114
115
116
117
118
119
120 OptixResult optixDeviceContextGetProperty(OptixDeviceContext context, OptixDeviceProperty property,
121 void* value, size_t sizeInBytes);
122
123
124
125
126
127
128
129
130
131
132
133
134
135 OptixResult optixDeviceContextSetLogCallback(OptixDeviceContext context,
136 OptixLogCallback callbackFunction,
137 void* callbackData,
138 unsigned int callbackLevel);
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158 OptixResult optixDeviceContextSetCacheEnabled(OptixDeviceContext context,
159 int enabled);
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216 OptixResult optixDeviceContextGetCacheEnabled(OptixDeviceContext context, int* enabled);
217
218
219
220
221
222
223 OptixResult optixDeviceContextGetCacheLocation(OptixDeviceContext context, char* location, size_t
224 locationSize);
225
226
227
228
229
230
231
232 OptixResult optixDeviceContextGetCacheDatabaseSizes(OptixDeviceContext context, size_t* lowWaterMark,
233 size_t* highWaterMark);
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262 OptixResult optixPipelineCreate(OptixDeviceContext context,
263 const OptixPipelineCompileOptions* pipelineCompileOptions,
264 const OptixPipelineLinkOptions* pipelineLinkOptions,
265 const OptixProgramGroup* programGroups,

```

```

266         unsigned int          numProgramGroups,
267         char*                  logString,
268         size_t*                 logStringSize,
269         OptixPipeline*          pipeline);
270
272 OptixResult optixPipelineDestroy(OptixPipeline pipeline);
273
296 OptixResult optixPipelineSetStackSize(OptixPipeline pipeline,
297         unsigned int directCallableStackSizeFromTraversal,
298         unsigned int directCallableStackSizeFromState,
299         unsigned int continuationStackSize,
300         unsigned int maxTraversableGraphDepth);
301
303
305
306
336 OptixResult optixModuleCreate(OptixDeviceContext context,
337         const OptixModuleCompileOptions* moduleCompileOptions,
338         const OptixPipelineCompileOptions* pipelineCompileOptions,
339         const char* input,
340         size_t inputSize,
341         char* logString,
342         size_t* logStringSize,
343         OptixModule* module);
344
364
377
385 OptixResult optixModuleCreateWithTasks(OptixDeviceContext context,
386         const OptixModuleCompileOptions* moduleCompileOptions,
387         const OptixPipelineCompileOptions* pipelineCompileOptions,
388         const char* input,
389         size_t inputSize,
390         char* logString,
391         size_t* logStringSize,
392         OptixModule* module,
393         OptixTask* firstTask);
394
401 OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState* state);
402
408 OptixResult optixModuleDestroy(OptixModule module);
409
413 OptixResult optixBuiltinISModuleGet(OptixDeviceContext context,
414         const OptixModuleCompileOptions* moduleCompileOptions,
415         const OptixPipelineCompileOptions* pipelineCompileOptions,
416         const OptixBuiltinISOptions* builtinISOptions,
417         OptixModule* builtinModule);
418
420
422
423
441 OptixResult optixTaskExecute(OptixTask task, OptixTask* additionalTasks, unsigned int
maxNumAdditionalTasks, unsigned int* numAdditionalTasksCreated);
442
444
446
447
456 OptixResult optixProgramGroupGetStackSize(OptixProgramGroup programGroup, OptixStackSizes* stackSizes,
OptixPipeline pipeline);
457
483 OptixResult optixProgramGroupCreate(OptixDeviceContext context,
484         const OptixProgramGroupDesc* programDescriptions,
485         unsigned int numProgramGroups,
486         const OptixProgramGroupOptions* options,
487         char* logString,
488         size_t* logStringSize,
489         OptixProgramGroup* programGroups);
490

```



```

492 OptixResult optixProgramGroupDestroy(OptixProgramGroup programGroup);
493
495
497
498
525 OptixResult optixLaunch(OptixPipeline          pipeline,
526                        CUstream                stream,
527                        CUdeviceptr              pipelineParams,
528                        size_t                   pipelineParamsSize,
529                        const OptixShaderBindingTable* sbt,
530                        unsigned int              width,
531                        unsigned int              height,
532                        unsigned int              depth);
533
536 OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void* sbtRecordHeaderHostPointer);
537
539
541
542
548 OptixResult optixAccelComputeMemoryUsage(OptixDeviceContext context,
549   const OptixAccelBuildOptions* accelOptions,
550   const OptixBuildInput* buildInputs,
551   unsigned int numBuildInputs,
552   OptixAccelBufferSizes* bufferSizes);
553
566 OptixResult optixAccelBuild(OptixDeviceContext context,
567                             CUstream          stream,
568                             const OptixAccelBuildOptions* accelOptions,
569                             const OptixBuildInput* buildInputs,
570                             unsigned int numBuildInputs,
571                             CUdeviceptr tempBuffer,
572                             size_t tempBufferSizeInBytes,
573                             CUdeviceptr outputBuffer,
574                             size_t outputBufferSizeInBytes,
575                             OptixTraversableHandle* outputHandle,
576                             const OptixAccelEmitDesc* emittedProperties,
577                             unsigned int numEmittedProperties);
578
596 OptixResult optixAccelGetRelocationInfo(OptixDeviceContext context, OptixTraversableHandle handle,
OptixRelocationInfo* info);
597
609 OptixResult optixCheckRelocationCompatibility(OptixDeviceContext context, const OptixRelocationInfo*
info, int* compatible);
610
648 OptixResult optixAccelRelocate(OptixDeviceContext context,
649                                CUstream          stream,
650                                const OptixRelocationInfo* info,
651                                const OptixRelocateInput* relocateInputs,
652                                size_t numRelocateInputs,
653                                CUdeviceptr targetAccel,
654                                size_t targetAccelSizeInBytes,
655                                OptixTraversableHandle* targetHandle);
656
674 OptixResult optixAccelCompact(OptixDeviceContext context,
675                               CUstream          stream,
676                               OptixTraversableHandle inputHandle,
677                               CUdeviceptr outputBuffer,
678                               size_t outputBufferSizeInBytes,
679                               OptixTraversableHandle* outputHandle);
680
689 OptixResult optixAccelEmitProperty(OptixDeviceContext context,
690                                    CUstream          stream,
691                                    OptixTraversableHandle handle,
692                                    const OptixAccelEmitDesc* emittedProperty);
693
698 OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext onDevice,
699  CUdeviceptr pointer,

```

```

700                                     OptixTraversableType    traversableType,
701                                     OptixTraversableHandle* traversableHandle);
702
703
709 OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext    context,
710   const OptixOpacityMicromapArrayBuildInput* buildInput,
711   OptixMicromapBufferSizes*    bufferSizes);
712
735 OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext    context,
736   CUstream            stream,
737   const OptixOpacityMicromapArrayBuildInput* buildInput,
738   const OptixMicromapBuffers*    buffers);
739
755 OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext context, CUdeviceptr
opacityMicromapArray, OptixRelocationInfo* info);
756
783 OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext    context,
784   CUstream            stream,
785   const OptixRelocationInfo* info,
786   CUdeviceptr          targetOpacityMicromapArray,
787   size_t
targetOpacityMicromapArraySizeInBytes);
788
794 OptixResult optixDisplacementMicromapArrayComputeMemoryUsage(OptixDeviceContext
context,
795   const
OptixDisplacementMicromapArrayBuildInput* buildInput,
796   OptixMicromapBufferSizes* bufferSizes);
797
811 OptixResult optixDisplacementMicromapArrayBuild(OptixDeviceContext    context,
812   CUstream            stream,
813   const OptixDisplacementMicromapArrayBuildInput* buildInput,
814   const OptixMicromapBuffers*    buffers);
815
816
818
820
821
833 OptixResult optixDenoiserCreate(OptixDeviceContext context,
834                                 OptixDenoiserModelKind modelKind,
835                                 const OptixDenoiserOptions* options,
836                                 OptixDenoiser* denoiser);
837
850 OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context,
851   const void* userData, size_t userDataSizeInBytes,
OptixDenoiser* denoiser);
852
854 OptixResult optixDenoiserDestroy(OptixDenoiser denoiser);
855
875 OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser denoiser,
876  unsigned int    outputWidth,
877  unsigned int    outputHeight,
878  OptixDenoiserSizes* returnSizes);
879
896 OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
897                                 CUstream      stream,
898                                 unsigned int  inputWidth,
899                                 unsigned int  inputHeight,
900                                 CUdeviceptr   denoiserState,
901                                 size_t        denoiserStateSizeInBytes,
902                                 CUdeviceptr   scratch,
903                                 size_t        scratchSizeInBytes);
904
970 OptixResult optixDenoiserInvoke(OptixDenoiser    denoiser,
971                                 CUstream          stream,
972                                 const OptixDenoiserParams* params,
973                                 CUdeviceptr        denoiserState,

```

```

974         size_t          denoiserStateSizeInBytes,
975         const OptixDenoiserGuideLayer* guideLayer,
976         const OptixDenoiserLayer* layers,
977         unsigned int numLayers,
978         unsigned int inputOffsetX,
979         unsigned int inputOffsetY,
980         CUdeviceptr scratch,
981         size_t scratchSizeInBytes);
982
1006 OptixResult optixDenoiserComputeIntensity(OptixDenoiser denoiser,
1007   CUstream stream,
1008   const OptixImage2D* inputImage,
1009   CUdeviceptr outputIntensity,
1010   CUdeviceptr scratch,
1011   size_t scratchSizeInBytes);
1012
1027 OptixResult optixDenoiserComputeAverageColor(OptixDenoiser denoiser,
1028  CUstream stream,
1029  const OptixImage2D* inputImage,
1030  CUdeviceptr outputAverageColor,
1031  CUdeviceptr scratch,
1032  size_t scratchSizeInBytes);
1033
1035
1036 #ifdef __cplusplus
1037 }
1038 #endif
1039
1040 #include "optix_function_table.h"
1041
1042 #endif // OPTIX_OPTIX_HOST_H

```

## 8.19 optix\_stack\_size.h File Reference

### Functions

- [OptixResult optixUtilAccumulateStackSizes](#) (OptixProgramGroup programGroup, OptixStackSizes \*stackSizes, OptixPipeline pipeline)
- [OptixResult optixUtilComputeStackSizes](#) (const OptixStackSizes \*stackSizes, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepth, unsigned int \*directCallableStackSizeFromTraversal, unsigned int \*directCallableStackSizeFromState, unsigned int \*continuationStackSize)
- [OptixResult optixUtilComputeStackSizesDCSplit](#) (const OptixStackSizes \*stackSizes, unsigned int dssDCFromTraversal, unsigned int dssDCFromState, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepthFromTraversal, unsigned int maxDCDepthFromState, unsigned int \*directCallableStackSizeFromTraversal, unsigned int \*directCallableStackSizeFromState, unsigned int \*continuationStackSize)
- [OptixResult optixUtilComputeStackSizesCssCCTree](#) (const OptixStackSizes \*stackSizes, unsigned int cssCCTree, unsigned int maxTraceDepth, unsigned int maxDCDepth, unsigned int \*directCallableStackSizeFromTraversal, unsigned int \*directCallableStackSizeFromState, unsigned int \*continuationStackSize)
- [OptixResult optixUtilComputeStackSizesSimplePathTracer](#) (OptixProgramGroup programGroupRG, OptixProgramGroup programGroupMS1, const OptixProgramGroup \*programGroupCH1, unsigned int programGroupCH1Count, OptixProgramGroup programGroupMS2, const OptixProgramGroup \*programGroupCH2, unsigned int programGroupCH2Count, unsigned int \*directCallableStackSizeFromTraversal, unsigned int \*directCallableStackSizeFromState, unsigned int \*continuationStackSize, OptixPipeline pipeline)

### 8.19.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

## 8.20 optix\_stack\_size.h

[Go to the documentation of this file.](#)

```

1 /*
2  * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
3  *
4  * Redistribution and use in source and binary forms, with or without
5  * modification, are permitted provided that the following conditions
6  * are met:
7  * * Redistributions of source code must retain the above copyright
8  *   notice, this list of conditions and the following disclaimer.
9  * * Redistributions in binary form must reproduce the above copyright
10 *   notice, this list of conditions and the following disclaimer in the
11 *   documentation and/or other materials provided with the distribution.
12 * * Neither the name of NVIDIA CORPORATION nor the names of its
13 *   contributors may be used to endorse or promote products derived
14 *   from this software without specific prior written permission.
15 *
16 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS "AS IS" AND ANY
17 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
18 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
19 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
20 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
21 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
22 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
23 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
24 * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
25 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
26 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
27 */
28
29
30
31
32
33 #ifndef OPTIX_OPTIX_STACK_SIZE_H
34 #define OPTIX_OPTIX_STACK_SIZE_H
35
36 #include "optix.h"
37
38 #include <algorithm>
39 #include <cstring>
40
41 #ifdef __cplusplus
42 extern "C" {
43 #endif
44
45 inline OptixResult optixUtilAccumulateStackSizes(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes, OptixPipeline pipeline)
46 {
47     if(!stackSizes)
48         return OPTIX_ERROR_INVALID_VALUE;
49
50     OptixStackSizes localStackSizes;
51     OptixResult result = optixProgramGroupGetStackSize(programGroup, &localStackSizes, pipeline);
52     if(result != OPTIX_SUCCESS)
53         return result;
54
55     stackSizes->cssRG = std::max(stackSizes->cssRG, localStackSizes.cssRG);
56     stackSizes->cssMS = std::max(stackSizes->cssMS, localStackSizes.cssMS);
57     stackSizes->cssCH = std::max(stackSizes->cssCH, localStackSizes.cssCH);
58 }
59
60
61
62
63
64
65
66

```

```

67     stackSizes->cssAH = std::max(stackSizes->cssAH, localStackSizes.cssAH);
68     stackSizes->cssIS = std::max(stackSizes->cssIS, localStackSizes.cssIS);
69     stackSizes->cssCC = std::max(stackSizes->cssCC, localStackSizes.cssCC);
70     stackSizes->dssDC = std::max(stackSizes->dssDC, localStackSizes.dssDC);
71
72     return OPTIX_SUCCESS;
73 }
74
75 inline OptixResult optixUtilComputeStackSizes(const OptixStackSizes* stackSizes,
76   unsigned int      maxTraceDepth,
77   unsigned int      maxCCDepth,
78   unsigned int      maxDCDepth,
79   unsigned int*     directCallableStackSizeFromTraversal,
80   unsigned int*     directCallableStackSizeFromState,
81   unsigned int*     continuationStackSize)
82 {
83     if(!stackSizes)
84         return OPTIX_ERROR_INVALID_VALUE;
85
86     const unsigned int cssRG = stackSizes->cssRG;
87     const unsigned int cssMS = stackSizes->cssMS;
88     const unsigned int cssCH = stackSizes->cssCH;
89     const unsigned int cssAH = stackSizes->cssAH;
90     const unsigned int cssIS = stackSizes->cssIS;
91     const unsigned int cssCC = stackSizes->cssCC;
92     const unsigned int dssDC = stackSizes->dssDC;
93
94     if(directCallableStackSizeFromTraversal)
95         *directCallableStackSizeFromTraversal = maxDCDepth * dssDC;
96     if(directCallableStackSizeFromState)
97         *directCallableStackSizeFromState = maxDCDepth * dssDC;
98
99     // upper bound on continuation stack used by call trees of continuation callables
100     unsigned int cssCCTree = maxCCDepth * cssCC;
101
102     // upper bound on continuation stack used by CH or MS programs including the call tree of
103     // continuation callables
104     unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
105
106     // clang-format off
107     if(continuationStackSize)
108         *continuationStackSize
109             = cssRG + cssCCTree
110               + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
111               + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
112     // clang-format on
113
114     return OPTIX_SUCCESS;
115 }
116
117 inline OptixResult optixUtilComputeStackSizesDCSplit(const OptixStackSizes* stackSizes,
118   unsigned int      dssDCFromTraversal,
119   unsigned int      dssDCFromState,
120   unsigned int      maxTraceDepth,
121   unsigned int      maxCCDepth,
122   unsigned int      maxDCDepthFromTraversal,
123   unsigned int      maxDCDepthFromState,
124   unsigned int*     directCallableStackSizeFromTraversal,
125   unsigned int*     directCallableStackSizeFromState,
126   unsigned int*     continuationStackSize)
127 {
128     if(!stackSizes)
129         return OPTIX_ERROR_INVALID_VALUE;
130
131     const unsigned int cssRG = stackSizes->cssRG;
132     const unsigned int cssMS = stackSizes->cssMS;

```

```

169     const unsigned int cssCH = stackSizes->cssCH;
170     const unsigned int cssAH = stackSizes->cssAH;
171     const unsigned int cssIS = stackSizes->cssIS;
172     const unsigned int cssCC = stackSizes->cssCC;
173     // use dssDCFromTraversal and dssDCFromState instead of stackSizes->dssDC
174
175     if(directCallableStackSizeFromTraversal)
176         *directCallableStackSizeFromTraversal = maxDCDepthFromTraversal * dssDCFromTraversal;
177     if(directCallableStackSizeFromState)
178         *directCallableStackSizeFromState = maxDCDepthFromState * dssDCFromState;
179
180     // upper bound on continuation stack used by call trees of continuation callables
181     unsigned int cssCCTree = maxCCDepth * cssCC;
182
183     // upper bound on continuation stack used by CH or MS programs including the call tree of
184     // continuation callables
185     unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
186
187     // clang-format off
188     if(continuationStackSize)
189         *continuationStackSize
190             = cssRG + cssCCTree
191             + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
192             + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
193     // clang-format on
194
195     return OPTIX_SUCCESS;
196 }
197
214 inline OptixResult optixUtilComputeStackSizesCssCCTree(const OptixStackSizes* stackSizes,
215   unsigned int          cssCCTree,
216   unsigned int          maxTraceDepth,
217   unsigned int          maxDCDepth,
218   unsigned int*
219 directCallableStackSizeFromTraversal,
220   unsigned int*          directCallableStackSizeFromState,
221   unsigned int*          continuationStackSize)
222 {
223     if(!stackSizes)
224         return OPTIX_ERROR_INVALID_VALUE;
225
226     const unsigned int cssRG = stackSizes->cssRG;
227     const unsigned int cssMS = stackSizes->cssMS;
228     const unsigned int cssCH = stackSizes->cssCH;
229     const unsigned int cssAH = stackSizes->cssAH;
230     const unsigned int cssIS = stackSizes->cssIS;
231     // use cssCCTree instead of stackSizes->cssCC and maxCCDepth
232     const unsigned int dssDC = stackSizes->dssDC;
233
234     if(directCallableStackSizeFromTraversal)
235         *directCallableStackSizeFromTraversal = maxDCDepth * dssDC;
236     if(directCallableStackSizeFromState)
237         *directCallableStackSizeFromState = maxDCDepth * dssDC;
238
239     // upper bound on continuation stack used by CH or MS programs including the call tree of
240     // continuation callables
241     unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
242
243     // clang-format off
244     if(continuationStackSize)
245         *continuationStackSize
246             = cssRG + cssCCTree
247             + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
248             + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
249     // clang-format on
250
251     return OPTIX_SUCCESS;

```

```

251 }
252
268 inline OptixResult optixUtilComputeStackSizesSimplePathTracer(OptixProgramGroup      programGroupRG,
269  OptixProgramGroup      programGroupMS1,
270  const OptixProgramGroup* programGroupCH1,
271  unsigned int              programGroupCH1Count,
272  OptixProgramGroup        programGroupMS2,
273  const OptixProgramGroup* programGroupCH2,
274  unsigned int              programGroupCH2Count,
275  unsigned int*
directCallableStackSizeFromTraversal,
276  unsigned int* directCallableStackSizeFromState,
277  unsigned int* continuationStackSize,
278  OptixPipeline pipeline)
279 {
280     if(!programGroupCH1 && (programGroupCH1Count > 0))
281         return OPTIX_ERROR_INVALID_VALUE;
282     if(!programGroupCH2 && (programGroupCH2Count > 0))
283         return OPTIX_ERROR_INVALID_VALUE;
284
285     OptixResult result;
286
287     OptixStackSizes stackSizesRG = {};
288     result                  = optixProgramGroupGetStackSize(programGroupRG, &stackSizesRG, pipeline);
289     if(result != OPTIX_SUCCESS)
290         return result;
291
292     OptixStackSizes stackSizesMS1 = {};
293     result                  = optixProgramGroupGetStackSize(programGroupMS1, &stackSizesMS1,
pipeline);
294     if(result != OPTIX_SUCCESS)
295         return result;
296
297     OptixStackSizes stackSizesCH1 = {};
298     for(unsigned int i = 0; i < programGroupCH1Count; ++i)
299     {
300         result = optixUtilAccumulateStackSizes(programGroupCH1[i], &stackSizesCH1, pipeline);
301         if(result != OPTIX_SUCCESS)
302             return result;
303     }
304
305     OptixStackSizes stackSizesMS2 = {};
306     result                  = optixProgramGroupGetStackSize(programGroupMS2, &stackSizesMS2,
pipeline);
307     if(result != OPTIX_SUCCESS)
308         return result;
309
310     OptixStackSizes stackSizesCH2 = {};
311     memset(&stackSizesCH2, 0, sizeof(OptixStackSizes));
312     for(unsigned int i = 0; i < programGroupCH2Count; ++i)
313     {
314         result = optixUtilAccumulateStackSizes(programGroupCH2[i], &stackSizesCH2, pipeline);
315         if(result != OPTIX_SUCCESS)
316             return result;
317     }
318
319     const unsigned int cssRG  = stackSizesRG.cssRG;
320     const unsigned int cssMS1 = stackSizesMS1.cssMS;
321     const unsigned int cssCH1 = stackSizesCH1.cssCH;
322     const unsigned int cssMS2 = stackSizesMS2.cssMS;
323     const unsigned int cssCH2 = stackSizesCH2.cssCH;
324     // no AH, IS, CC, or DC programs
325
326     if(directCallableStackSizeFromTraversal)
327         *directCallableStackSizeFromTraversal = 0;
328     if(directCallableStackSizeFromState)
329         *directCallableStackSizeFromState = 0;

```



```

330
331     if(continuationStackSize)
332         *continuationStackSize = cssRG + std::max(cssMS1, cssCH1 + std::max(cssMS2, cssCH2));
333
334     return OPTIX_SUCCESS;
335 }
336 // end group optix_utilities
337
338
339 #ifdef __cplusplus
340 }
341 #endif
342
343 #endif // OPTIX_OPTIX_STACK_SIZE_H

```

## 8.21 optix\_stubs.h File Reference

### Macros

- `#define WIN32_LEAN_AND_MEAN 1`

### Functions

- `static void * optixLoadWindowsDllFromName (const char *optixDllName)`
- `static void * optixLoadWindowsDll ()`
- `OptixResult optixInitWithHandle (void **handlePtr)`
- `OptixResult optixInit (void)`
- `OptixResult optixUninitWithHandle (void *handle)`

### Variables

- `OptixFunctionTable g_optixFunctionTable`

#### 8.21.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

#### 8.21.2 Macro Definition Documentation

##### 8.21.2.1 WIN32\_LEAN\_AND\_MEAN

```
#define WIN32_LEAN_AND_MEAN 1
```

#### 8.21.3 Function Documentation

##### 8.21.3.1 optixLoadWindowsDll()

```
static void * optixLoadWindowsDll ( ) [static]
```

##### 8.21.3.2 optixLoadWindowsDllFromName()

```
static void * optixLoadWindowsDllFromName (
    const char * optixDllName ) [static]
```



## 8.22 optix\_stubs.h

[Go to the documentation of this file.](#)

```

1 /*
2 * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
3 *
4 * Redistribution and use in source and binary forms, with or without
5 * modification, are permitted provided that the following conditions
6 * are met:
7 * * Redistributions of source code must retain the above copyright
8 *   notice, this list of conditions and the following disclaimer.
9 * * Redistributions in binary form must reproduce the above copyright
10 *   notice, this list of conditions and the following disclaimer in the
11 *   documentation and/or other materials provided with the distribution.
12 * * Neither the name of NVIDIA CORPORATION nor the names of its
13 *   contributors may be used to endorse or promote products derived
14 *   from this software without specific prior written permission.
15 *
16 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS "AS IS" AND ANY
17 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
18 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
19 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
20 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
21 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
22 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
23 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
24 * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
25 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
26 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
27 */
28
29
30
31
32
33 #ifndef OPTIX_OPTIX_STUBS_H
34 #define OPTIX_OPTIX_STUBS_H
35
36 #include "optix_function_table.h"
37
38 #ifdef _WIN32
39 #ifndef WIN32_LEAN_AND_MEAN
40 #define WIN32_LEAN_AND_MEAN 1
41 #endif
42 #include <windows.h>
43 // The cfgmgr32 header is necessary for interrogating driver information in the registry.
44 // For convenience the library is also linked in automatically using the #pragma command.
45 #include <cfgmgr32.h>
46 #pragma comment(lib, "Cfgmgr32.lib")
47 #include <string.h>
48 #else
49 #include <dlfcn.h>
50 #endif
51
52 #ifdef __cplusplus
53 extern "C" {
54 #endif
55
56 // The function table needs to be defined in exactly one translation unit. This can be
57 // achieved by including optix_function_table_definition.h in that translation unit.
58 extern OptixFunctionTable g_optixFunctionTable;
59
60 #ifdef _WIN32
61 #if defined(_MSC_VER)
62 // Visual Studio produces warnings suggesting strcpy and friends being replaced with _s
63 // variants. All the string lengths and allocation sizes have been calculated and should
64 // be safe, so we are disabling this warning to increase compatibility.
65 #pragma warning(push)
66 #pragma warning(disable : 4996)
67 #endif

```

```

68 static void* optixLoadWindowsDllFromName(const char* optixDllName)
69 {
70     void* handle = NULL;
71
72     // Try the bare dll name first. This picks it up in the local path, followed by
73     // standard Windows paths.
74     handle = LoadLibraryA((LPSTR)optixDllName);
75     if(handle)
76         return handle;
77 // If we don't find it in the default dll search path, try the system paths
78
79     // Get the size of the path first, then allocate
80     unsigned int size = GetSystemDirectoryA(NULL, 0);
81     if(size == 0)
82     {
83         // Couldn't get the system path size, so bail
84         return NULL;
85     }
86     size_t pathSize = size + 1 + strlen(optixDllName);
87     char* systemPath = (char*)malloc(pathSize);
88     if(systemPath == NULL)
89         return NULL;
90     if(GetSystemDirectoryA(systemPath, size) != size - 1)
91     {
92         // Something went wrong
93         free(systemPath);
94         return NULL;
95     }
96     strcat(systemPath, "\\");
97     strcat(systemPath, optixDllName);
98     handle = LoadLibraryA(systemPath);
99     free(systemPath);
100     if(handle)
101         return handle;
102
103     // If we didn't find it, go looking in the register store. Since nvoptix.dll doesn't
104     // have its own registry entry, we are going to look for the opengl driver which lives
105     // next to nvoptix.dll. 0 (null) will be returned if any errors occurred.
106
107     static const char* deviceInstanceIdentifiersGUID = "{4d36e968-e325-11ce-bfc1-08002be10318}";
108     const ULONG flags = CM_GETIDLIST_FILTER_CLASS |
CM_GETIDLIST_FILTER_PRESENT;
109     ULONG deviceListSize = 0;
110     if(CM_Get_Device_ID_List_SizeA(&deviceListSize, deviceInstanceIdentifiersGUID, flags) != CR_SUCCESS)
111     {
112         return NULL;
113     }
114     char* deviceNames = (char*)malloc(deviceListSize);
115     if(deviceNames == NULL)
116         return NULL;
117     if(CM_Get_Device_ID_ListA(deviceInstanceIdentifiersGUID, deviceNames, deviceListSize, flags))
118     {
119         free(deviceNames);
120         return NULL;
121     }
122     DEVINST devID = 0;
123     char* dllPath = NULL;
124
125     // Continue to the next device if errors are encountered.
126     for(char* deviceName = deviceNames; *deviceName; deviceName += strlen(deviceName) + 1)
127     {
128         if(CM_Locate_DevNodeA(&devID, deviceName, CM_LOCATE_DEVNODE_NORMAL) != CR_SUCCESS)
129         {
130             continue;
131         }
132         HKEY regKey = 0;
133         if(CM_Open_DevNode_Key(devID, KEY_QUERY_VALUE, 0, RegDisposition_OpenExisting, &regKey,

```

```

CM_REGISTRY_SOFTWARE) != CR_SUCCESS)
134     {
135         continue;
136     }
137     const char* valueName = "OpenGLDriverName";
138     DWORD      valueSize = 0;
139     LSTATUS    ret       = RegQueryValueExA(regKey, valueName, NULL, NULL, NULL, &valueSize);
140     if(ret != ERROR_SUCCESS)
141     {
142         RegCloseKey(regKey);
143         continue;
144     }
145     char* regValue = (char*)malloc(valueSize);
146     if(regValue == NULL)
147     {
148         RegCloseKey(regKey);
149         continue;
150     }
151     ret = RegQueryValueExA(regKey, valueName, NULL, NULL, (LPBYTE)regValue, &valueSize);
152     if(ret != ERROR_SUCCESS)
153     {
154         free(regValue);
155         RegCloseKey(regKey);
156         continue;
157     }
158     // Strip the opengl driver dll name from the string then create a new string with
159     // the path and the nvoptix.dll name
160     for(int i = (int) valueSize - 1; i >= 0 && regValue[i] != '\\'; --i)
161         regValue[i] = '\\0';
162     size_t newPathSize = strlen(regValue) + strlen(optixDllName) + 1;
163     dllPath = (char*)malloc(newPathSize);
164     if(dllPath == NULL)
165     {
166         free(regValue);
167         RegCloseKey(regKey);
168         continue;
169     }
170     strcpy(dllPath, regValue);
171     strcat(dllPath, optixDllName);
172     free(regValue);
173     RegCloseKey(regKey);
174     handle = LoadLibraryA((LPCSTR)dllPath);
175     free(dllPath);
176     if(handle)
177         break;
178     }
179     free(deviceNames);
180     return handle;
181 }
182 #if defined(_MSC_VER)
183 #pragma warning(pop)
184 #endif
185
186 static void* optixLoadWindowsDll( )
187 {
188     return optixLoadWindowsDllFromName("nvoptix.dll");
189 }
190 #endif
191
192
204 inline OptixResult optixInitWithHandle(void** handlePtr)
205 {
206     // Make sure these functions get initialized to zero in case the DLL and function
207     // table can't be loaded
208     g_optixFunctionTable.optixGetErrorName = 0;
209     g_optixFunctionTable.optixGetErrorString = 0;
210

```

```

211     if(!handlePtr)
212         return OPTIX_ERROR_INVALID_VALUE;
213
214 #ifdef _WIN32
215     *handlePtr = optixLoadWindowsDll();
216     if(!*handlePtr)
217         return OPTIX_ERROR_LIBRARY_NOT_FOUND;
218
219     void* symbol = GetProcAddress((HMODULE)*handlePtr, "optixQueryFunctionTable");
220     if(!symbol)
221         return OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND;
222 #else
223     *handlePtr = dlopen("libnvoptix.so.1", RTLD_NOW);
224     if(!*handlePtr)
225         return OPTIX_ERROR_LIBRARY_NOT_FOUND;
226
227     void* symbol = dlsym(*handlePtr, "optixQueryFunctionTable");
228     if(!symbol)
229         return OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND;
230 #endif
231
232     OptixQueryFunctionTable_t* optixQueryFunctionTable = (OptixQueryFunctionTable_t*)symbol;
233
234     return optixQueryFunctionTable(OPTIX_ABI_VERSION, 0, 0, 0, &g_optixFunctionTable,
235     sizeof(g_optixFunctionTable));
236 }
237
240 inline OptixResult optixInit(void)
241 {
242     void* handle;
243     return optixInitWithHandle(&handle);
244 }
245
251 inline OptixResult optixUninitWithHandle(void* handle)
252 {
253     if(!handle)
254         return OPTIX_ERROR_INVALID_VALUE;
255 #ifdef _WIN32
256     if(!FreeLibrary((HMODULE)handle))
257         return OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE;
258 #else
259     if(dlclose(handle))
260         return OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE;
261 #endif
262     OptixFunctionTable empty = { 0 };
263     g_optixFunctionTable = empty;
264     return OPTIX_SUCCESS;
265 }
266
267 // end group optix_utilities
269
270 #ifndef OPTIX_DOXYGEN_SHOULD_SKIP_THIS
271
272 // Stub functions that forward calls to the corresponding function pointer in the function table.
273
274 inline const char* optixGetErrorName(OptixResult result)
275 {
276     if(g_optixFunctionTable.optixGetErrorName)
277         return g_optixFunctionTable.optixGetErrorName(result);
278
279     // If the DLL and symbol table couldn't be loaded, provide a set of error strings
280     // suitable for processing errors related to the DLL loading.
281     switch(result)
282     {
283     case OPTIX_SUCCESS:
284         return "OPTIX_SUCCESS";
285     case OPTIX_ERROR_INVALID_VALUE:

```

```

286         return "OPTIX_ERROR_INVALID_VALUE";
287     case OPTIX_ERROR_UNSUPPORTED_ABI_VERSION:
288         return "OPTIX_ERROR_UNSUPPORTED_ABI_VERSION";
289     case OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH:
290         return "OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH";
291     case OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS:
292         return "OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS";
293     case OPTIX_ERROR_LIBRARY_NOT_FOUND:
294         return "OPTIX_ERROR_LIBRARY_NOT_FOUND";
295     case OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND:
296         return "OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND";
297     case OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE:
298         return "OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE";
299     default:
300         return "Unknown OptixResult code";
301     }
302 }
303
304 inline const char* optixGetErrorString(OptixResult result)
305 {
306     if(g_optixFunctionTable.optixGetErrorString)
307         return g_optixFunctionTable.optixGetErrorString(result);
308
309     // If the DLL and symbol table couldn't be loaded, provide a set of error strings
310     // suitable for processing errors related to the DLL loading.
311     switch(result)
312     {
313     case OPTIX_SUCCESS:
314         return "Success";
315     case OPTIX_ERROR_INVALID_VALUE:
316         return "Invalid value";
317     case OPTIX_ERROR_UNSUPPORTED_ABI_VERSION:
318         return "Unsupported ABI version";
319     case OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH:
320         return "Function table size mismatch";
321     case OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS:
322         return "Invalid options to entry function";
323     case OPTIX_ERROR_LIBRARY_NOT_FOUND:
324         return "Library not found";
325     case OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND:
326         return "Entry symbol not found";
327     case OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE:
328         return "Library could not be unloaded";
329     default:
330         return "Unknown OptixResult code";
331     }
332 }
333
334 inline OptixResult optixDeviceContextCreate(CUcontext fromContext, const OptixDeviceContextOptions*
options, OptixDeviceContext* context)
335 {
336     return g_optixFunctionTable.optixDeviceContextCreate(fromContext, options, context);
337 }
338
339 inline OptixResult optixDeviceContextDestroy(OptixDeviceContext context)
340 {
341     return g_optixFunctionTable.optixDeviceContextDestroy(context);
342 }
343
344 inline OptixResult optixDeviceContextGetProperty(OptixDeviceContext context, OptixDeviceProperty
property, void* value, size_t sizeInBytes)
345 {
346     return g_optixFunctionTable.optixDeviceContextGetProperty(context, property, value, sizeInBytes);
347 }
348
349 inline OptixResult optixDeviceContextSetLogCallback(OptixDeviceContext context,
OptixLogCallback callbackFunction,
350

```

```

351                                     void*          callbackData,
352                                     unsigned int    callbackLevel)
353 {
354     return g_optixFunctionTable.optixDeviceContextSetLogCallback(context, callbackFunction,
355     callbackData, callbackLevel);
356 }
357 inline OptixResult optixDeviceContextSetCacheEnabled(OptixDeviceContext context, int enabled)
358 {
359     return g_optixFunctionTable.optixDeviceContextSetCacheEnabled(context, enabled);
360 }
361
362 inline OptixResult optixDeviceContextSetCacheLocation(OptixDeviceContext context, const char* location)
363 {
364     return g_optixFunctionTable.optixDeviceContextSetCacheLocation(context, location);
365 }
366
367 inline OptixResult optixDeviceContextSetCacheDatabaseSizes(OptixDeviceContext context, size_t
368 lowWaterMark, size_t highWaterMark)
369 {
370     return g_optixFunctionTable.optixDeviceContextSetCacheDatabaseSizes(context, lowWaterMark,
371     highWaterMark);
372 }
373
374 inline OptixResult optixDeviceContextGetCacheEnabled(OptixDeviceContext context, int* enabled)
375 {
376     return g_optixFunctionTable.optixDeviceContextGetCacheEnabled(context, enabled);
377 }
378
379 inline OptixResult optixDeviceContextGetCacheLocation(OptixDeviceContext context, char* location, size_t
380 locationSize)
381 {
382     return g_optixFunctionTable.optixDeviceContextGetCacheLocation(context, location, locationSize);
383 }
384
385 inline OptixResult optixDeviceContextGetCacheDatabaseSizes(OptixDeviceContext context, size_t*
386 lowWaterMark, size_t* highWaterMark)
387 {
388     return g_optixFunctionTable.optixDeviceContextGetCacheDatabaseSizes(context, lowWaterMark,
389     highWaterMark);
390 }
391
392 inline OptixResult optixModuleCreate(OptixDeviceContext context,
393                                     const OptixModuleCompileOptions* moduleCompileOptions,
394                                     const OptixPipelineCompileOptions* pipelineCompileOptions,
395                                     const char* input,
396                                     size_t inputSize,
397                                     char* logString,
398                                     size_t* logStringSize,
399                                     OptixModule* module)
400 {
401     return g_optixFunctionTable.optixModuleCreate(context, moduleCompileOptions, pipelineCompileOptions,
402     input, inputSize,
403     logString, logStringSize, module);
404 }
405
406 inline OptixResult optixModuleCreateWithTasks(OptixDeviceContext context,
407                                     const OptixModuleCompileOptions* moduleCompileOptions,
408                                     const OptixPipelineCompileOptions* pipelineCompileOptions,
409                                     const char* input,
410                                     size_t inputSize,
411                                     char* logString,
412                                     size_t* logStringSize,
413                                     OptixModule* module,
414                                     OptixTask* firstTask)
415 {
416     return g_optixFunctionTable.optixModuleCreateWithTasks(context, moduleCompileOptions,

```

```

pipelineCompileOptions, input,
411                                     inputSize, logString, logStringSize, module,
firstTask);
412 }
413
414 inline OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState* state)
415 {
416     return g_optixFunctionTable.optixModuleGetCompilationState(module, state);
417 }
418
419 inline OptixResult optixModuleDestroy(OptixModule module)
420 {
421     return g_optixFunctionTable.optixModuleDestroy(module);
422 }
423
424 inline OptixResult optixBuiltinISModuleGet(OptixDeviceContext context,
425                                     const OptixModuleCompileOptions* moduleCompileOptions,
426                                     const OptixPipelineCompileOptions* pipelineCompileOptions,
427                                     const OptixBuiltinISOptions* builtinISOptions,
428                                     OptixModule* builtinModule)
429 {
430     return g_optixFunctionTable.optixBuiltinISModuleGet(context, moduleCompileOptions,
431 pipelineCompileOptions,
432                                     builtinISOptions, builtinModule);
433 }
434 inline OptixResult optixTaskExecute(OptixTask task, OptixTask* additionalTasks, unsigned int
435 maxNumAdditionalTasks, unsigned int* numAdditionalTasksCreated)
436 {
437     return g_optixFunctionTable.optixTaskExecute(task, additionalTasks, maxNumAdditionalTasks,
438 numAdditionalTasksCreated);
439 }
440
441 inline OptixResult optixProgramGroupCreate(OptixDeviceContext context,
442                                     const OptixProgramGroupDesc* programDescriptions,
443                                     unsigned int numProgramGroups,
444                                     const OptixProgramGroupOptions* options,
445                                     char* logString,
446                                     size_t* logStringSize,
447                                     OptixProgramGroup* programGroups)
448 {
449     return g_optixFunctionTable.optixProgramGroupCreate(context, programDescriptions, numProgramGroups,
450 options,
451                                     logString, logStringSize, programGroups);
452 }
453
454 inline OptixResult optixProgramGroupDestroy(OptixProgramGroup programGroup)
455 {
456     return g_optixFunctionTable.optixProgramGroupDestroy(programGroup);
457 }
458
459 inline OptixResult optixProgramGroupGetStackSize(OptixProgramGroup programGroup, OptixStackSizes*
460 stackSizes, OptixPipeline pipeline)
461 {
462     return g_optixFunctionTable.optixProgramGroupGetStackSize(programGroup, stackSizes, pipeline);
463 }
464
465 inline OptixResult optixPipelineCreate(OptixDeviceContext context,
466                                     const OptixPipelineCompileOptions* pipelineCompileOptions,
467                                     const OptixPipelineLinkOptions* pipelineLinkOptions,
468                                     const OptixProgramGroup* programGroups,
469                                     unsigned int numProgramGroups,
470                                     char* logString,
471                                     size_t* logStringSize,
472                                     OptixPipeline* pipeline)
473 {
474     return g_optixFunctionTable.optixPipelineCreate(context, pipelineCompileOptions,

```

```

pipelineLinkOptions, programGroups,
471                                     numProgramGroups, logString, logStringSize, pipeline);
472 }
473
474 inline OptixResult optixPipelineDestroy(OptixPipeline pipeline)
475 {
476     return g_optixFunctionTable.optixPipelineDestroy(pipeline);
477 }
478
479 inline OptixResult optixPipelineSetStackSize(OptixPipeline pipeline,
480   unsigned int directCallableStackSizeFromTraversal,
481   unsigned int directCallableStackSizeFromState,
482   unsigned int continuationStackSize,
483   unsigned int maxTraversableGraphDepth)
484 {
485     return g_optixFunctionTable.optixPipelineSetStackSize(pipeline,
486 directCallableStackSizeFromTraversal, directCallableStackSizeFromState,
487                                     continuationStackSize, maxTraversableGraphDepth);
488 }
489
490 inline OptixResult optixAccelComputeMemoryUsage(OptixDeviceContext context,
491   const OptixAccelBuildOptions* accelOptions,
492   const OptixBuildInput* buildInputs,
493   unsigned int numBuildInputs,
494   OptixAccelBufferSizes* bufferSizes)
495 {
496     return g_optixFunctionTable.optixAccelComputeMemoryUsage(context, accelOptions, buildInputs,
497 numBuildInputs, bufferSizes);
498 }
499
500 inline OptixResult optixAccelBuild(OptixDeviceContext context,
501                                    CUstream stream,
502                                    const OptixAccelBuildOptions* accelOptions,
503                                    const OptixBuildInput* buildInputs,
504                                    unsigned int numBuildInputs,
505                                    CUdeviceptr tempBuffer,
506                                    size_t tempBufferSizeInBytes,
507                                    CUdeviceptr outputBuffer,
508                                    size_t outputBufferSizeInBytes,
509                                    OptixTraversableHandle* outputHandle,
510                                    const OptixAccelEmitDesc* emittedProperties,
511                                    unsigned int numEmittedProperties)
512 {
513     return g_optixFunctionTable.optixAccelBuild(context, stream, accelOptions, buildInputs,
514 numBuildInputs, tempBuffer,
515                                     tempBufferSizeInBytes, outputBuffer, outputBufferSizeInBytes,
516                                     outputHandle, emittedProperties, numEmittedProperties);
517 }
518
519 inline OptixResult optixAccelGetRelocationInfo(OptixDeviceContext context, OptixTraversableHandle
520 handle, OptixRelocationInfo* info)
521 {
522     return g_optixFunctionTable.optixAccelGetRelocationInfo(context, handle, info);
523 }
524
525 inline OptixResult optixCheckRelocationCompatibility(OptixDeviceContext context, const
526 OptixRelocationInfo* info, int* compatible)
527 {
528     return g_optixFunctionTable.optixCheckRelocationCompatibility(context, info, compatible);
529 }
530
531 inline OptixResult optixAccelRelocate(OptixDeviceContext context,
532                                       CUstream stream,
533                                       const OptixRelocationInfo* info,
534                                       const OptixRelocateInput* relocateInputs,

```



```

532                                     size_t                numRelocateInputs,
533                                     CUdeviceptr            targetAccel,
534                                     size_t                targetAccelSizeInBytes,
535                                     OptixTraversableHandle* targetHandle)
536 {
537     return g_optixFunctionTable.optixAccelRelocate(context, stream, info, relocateInputs,
numRelocateInputs,
538   targetAccel, targetAccelSizeInBytes, targetHandle);
539 }
540
541 inline OptixResult optixAccelCompact(OptixDeviceContext context,
542                                     CUstream          stream,
543                                     OptixTraversableHandle inputHandle,
544                                     CUdeviceptr          outputBuffer,
545                                     size_t              outputBufferSizeInBytes,
546                                     OptixTraversableHandle* outputHandle)
547 {
548     return g_optixFunctionTable.optixAccelCompact(context, stream, inputHandle, outputBuffer,
outputBufferSizeInBytes, outputHandle);
549 }
550
551 inline OptixResult optixAccelEmitProperty(OptixDeviceContext context,
552                                     CUstream          stream,
553                                     OptixTraversableHandle handle,
554                                     const OptixAccelEmitDesc* emittedProperty)
555 {
556     return g_optixFunctionTable.optixAccelEmitProperty(context, stream, handle, emittedProperty);
557 }
558
559 inline OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext onDevice,
560                                     CUdeviceptr          pointer,
561                                     OptixTraversableType traversableType,
562                                     OptixTraversableHandle* traversableHandle)
563 {
564     return g_optixFunctionTable.optixConvertPointerToTraversableHandle(onDevice, pointer,
traversableType, traversableHandle);
565 }
566
567 inline OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext
context,
568                                     const OptixOpacityMicromapArrayBuildInput*
buildInput,
569                                     OptixMicromapBufferSizes*
bufferSizes)
570 {
571     return g_optixFunctionTable.optixOpacityMicromapArrayComputeMemoryUsage(context, buildInput,
bufferSizes);
572 }
573
574 inline OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext context,
575                                     CUstream          stream,
576                                     const OptixOpacityMicromapArrayBuildInput* buildInput,
577                                     const OptixMicromapBuffers* buffers)
578 {
579     return g_optixFunctionTable.optixOpacityMicromapArrayBuild(context, stream, buildInput, buffers);
580 }
581
582 inline OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext context,
583                                     CUdeviceptr          opacityMicromapArray,
584                                     OptixRelocationInfo* info)
585 {
586     return g_optixFunctionTable.optixOpacityMicromapArrayGetRelocationInfo(context,
opacityMicromapArray, info);
587 }
588
589 inline OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext context,
590                                     CUstream          stream,

```

```

591                                     const OptixRelocationInfo* info,
592                                     CUdeviceptr                targetOpacityMicromapArray,
593                                     size_t                      size_t
targetOpacityMicromapArraySizeInBytes)
594 {
595     return g_optixFunctionTable.optixOpacityMicromapArrayRelocate(context, stream, info,
targetOpacityMicromapArray, targetOpacityMicromapArraySizeInBytes);
596 }
597
598 inline OptixResult optixDisplacementMicromapArrayComputeMemoryUsage(OptixDeviceContext context,
599                                     const
OptixDisplacementMicromapArrayBuildInput* buildInput,
600                                     OptixMicromapBufferSizes* bufferSizes)
601 {
602     return g_optixFunctionTable.optixDisplacementMicromapArrayComputeMemoryUsage(context, buildInput,
bufferSizes);
603 }
604
605 inline OptixResult optixDisplacementMicromapArrayBuild(OptixDeviceContext
context,
606                                     CUstream                                stream,
607                                     const OptixDisplacementMicromapArrayBuildInput*
buildInput,
608                                     const OptixMicromapBuffers*                buffers)
609 {
610     return g_optixFunctionTable.optixDisplacementMicromapArrayBuild(context, stream, buildInput,
buffers);
611 }
612
613 inline OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer)
614 {
615     return g_optixFunctionTable.optixSbtRecordPackHeader(programGroup, sbtRecordHeaderHostPointer);
616 }
617
618 inline OptixResult optixLaunch(OptixPipeline                pipeline,
619                                     CUstream                stream,
620                                     CUdeviceptr              pipelineParams,
621                                     size_t                    pipelineParamsSize,
622                                     const OptixShaderBindingTable* sbt,
623                                     unsigned int              width,
624                                     unsigned int              height,
625                                     unsigned int              depth)
626 {
627     return g_optixFunctionTable.optixLaunch(pipeline, stream, pipelineParams, pipelineParamsSize, sbt,
width, height, depth);
628 }
629
630 inline OptixResult optixDenoiserCreate(OptixDeviceContext context, OptixDenoiserModelKind modelKind,
const OptixDenoiserOptions* options, OptixDenoiser* returnHandle)
631 {
632     return g_optixFunctionTable.optixDenoiserCreate(context, modelKind, options, returnHandle);
633 }
634
635 inline OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context, const void* data, size_t
dataSizeInBytes, OptixDenoiser* returnHandle)
636 {
637     return g_optixFunctionTable.optixDenoiserCreateWithUserModel(context, data, dataSizeInBytes,
returnHandle);
638 }
639
640 inline OptixResult optixDenoiserDestroy(OptixDenoiser handle)
641 {
642     return g_optixFunctionTable.optixDenoiserDestroy(handle);
643 }
644
645 inline OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser handle,

```

```

646                                     unsigned int      maximumInputWidth,
647                                     unsigned int      maximumInputHeight,
648                                     OptixDenoiserSizes* returnSizes)
649 {
650     return g_optixFunctionTable.optixDenoiserComputeMemoryResources(handle, maximumInputWidth,
651     maximumInputHeight, returnSizes);
652 }
653 inline OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
654                                     CUstream      stream,
655                                     unsigned int   inputWidth,
656                                     unsigned int   inputHeight,
657                                     CUdeviceptr    denoiserState,
658                                     size_t         denoiserStateSizeInBytes,
659                                     CUdeviceptr    scratch,
660                                     size_t         scratchSizeInBytes)
661 {
662     return g_optixFunctionTable.optixDenoiserSetup(denoiser, stream, inputWidth, inputHeight,
663     denoiserState,
664                                     denoiserStateSizeInBytes, scratch, scratchSizeInBytes);
665 }
666 inline OptixResult optixDenoiserInvoke(OptixDenoiser handle,
667                                     CUstream      stream,
668                                     const OptixDenoiserParams* params,
669                                     CUdeviceptr    denoiserData,
670                                     size_t         denoiserDataSize,
671                                     const OptixDenoiserGuideLayer* guideLayer,
672                                     const OptixDenoiserLayer* layers,
673                                     unsigned int   numLayers,
674                                     unsigned int   inputOffsetX,
675                                     unsigned int   inputOffsetY,
676                                     CUdeviceptr    scratch,
677                                     size_t         scratchSizeInBytes)
678 {
679     return g_optixFunctionTable.optixDenoiserInvoke(handle, stream, params, denoiserData,
680     denoiserDataSize,
681                                     guidelayer, layers, numLayers,
682     inputOffsetX, inputOffsetY, scratch, scratchSizeInBytes);
683 }
684 inline OptixResult optixDenoiserComputeIntensity(OptixDenoiser handle,
685                                     CUstream      stream,
686                                     const OptixImage2D* inputImage,
687                                     CUdeviceptr    outputIntensity,
688                                     CUdeviceptr    scratch,
689                                     size_t         scratchSizeInBytes)
690 {
691     return g_optixFunctionTable.optixDenoiserComputeIntensity(handle, stream, inputImage,
692     outputIntensity, scratch, scratchSizeInBytes);
693 }
694 inline OptixResult optixDenoiserComputeAverageColor(OptixDenoiser handle,
695                                     CUstream      stream,
696                                     const OptixImage2D* inputImage,
697                                     CUdeviceptr    outputAverageColor,
698                                     CUdeviceptr    scratch,
699                                     size_t         scratchSizeInBytes)
700 {
701     return g_optixFunctionTable.optixDenoiserComputeAverageColor(handle, stream, inputImage,
702     outputAverageColor, scratch, scratchSizeInBytes);
703 }
704 #endif // OPTIX_DOXYGEN_SHOULD_SKIP_THIS
705
706 #ifdef __cplusplus
707 }

```

```

708 #endif
709
710 #endif // OPTIX_OPTIX_STUBS_H

```

## 8.23 optix\_types.h File Reference

### Classes

- struct OptixDeviceContextOptions
- struct OptixOpacityMicromapUsageCount
- struct OptixBuildInputOpacityMicromap
- struct OptixRelocateInputOpacityMicromap
- struct OptixDisplacementMicromapDesc
- struct OptixDisplacementMicromapHistogramEntry
- struct OptixDisplacementMicromapArrayBuildInput
- struct OptixDisplacementMicromapUsageCount
- struct OptixBuildInputDisplacementMicromap
- struct OptixBuildInputTriangleArray
- struct OptixRelocateInputTriangleArray
- struct OptixBuildInputCurveArray
- struct OptixBuildInputSphereArray
- struct OptixAabb
- struct OptixBuildInputCustomPrimitiveArray
- struct OptixBuildInputInstanceArray
- struct OptixRelocateInputInstanceArray
- struct OptixBuildInput
- struct OptixRelocateInput
- struct OptixInstance
- struct OptixOpacityMicromapDesc
- struct OptixOpacityMicromapHistogramEntry
- struct OptixOpacityMicromapArrayBuildInput
- struct OptixMicromapBufferSizes
- struct OptixMicromapBuffers
- struct OptixMotionOptions
- struct OptixAccelBuildOptions
- struct OptixAccelBufferSizes
- struct OptixAccelEmitDesc
- struct OptixRelocationInfo
- struct OptixStaticTransform
- struct OptixMatrixMotionTransform
- struct OptixSRTData
- struct OptixSRTMotionTransform
- struct OptixImage2D
- struct OptixDenoiserOptions
- struct OptixDenoiserGuideLayer
- struct OptixDenoiserLayer
- struct OptixDenoiserParams
- struct OptixDenoiserSizes
- struct OptixModuleCompileBoundValueEntry
- struct OptixPayloadType
- struct OptixModuleCompileOptions

- struct OptixProgramGroupSingleModule
- struct OptixProgramGroupHitgroup
- struct OptixProgramGroupCallables
- struct OptixProgramGroupDesc
- struct OptixProgramGroupOptions
- struct OptixPipelineCompileOptions
- struct OptixPipelineLinkOptions
- struct OptixShaderBindingTable
- struct OptixStackSizes
- struct OptixBuiltinISOOptions

## Macros

- #define OPTIX\_SBT\_RECORD\_HEADER\_SIZE ((size\_t)32)
- #define OPTIX\_SBT\_RECORD\_ALIGNMENT 16ull
- #define OPTIX\_ACCEL\_BUFFER\_BYTE\_ALIGNMENT 128ull
- #define OPTIX\_INSTANCE\_BYTE\_ALIGNMENT 16ull
- #define OPTIX\_AABB\_BUFFER\_BYTE\_ALIGNMENT 8ull
- #define OPTIX\_GEOMETRY\_TRANSFORM\_BYTE\_ALIGNMENT 16ull
- #define OPTIX\_TRANSFORM\_BYTE\_ALIGNMENT 64ull
- #define OPTIX\_OPACITY\_MICROMAP\_DESC\_BUFFER\_BYTE\_ALIGNMENT 8ull
- #define OPTIX\_COMPILE\_DEFAULT\_MAX\_REGISTER\_COUNT 0
- #define OPTIX\_COMPILE\_DEFAULT\_MAX\_PAYLOAD\_TYPE\_COUNT 8
- #define OPTIX\_COMPILE\_DEFAULT\_MAX\_PAYLOAD\_VALUE\_COUNT 32
- #define OPTIX\_OPACITY\_MICROMAP\_STATE\_TRANSPARENT (0)
- #define OPTIX\_OPACITY\_MICROMAP\_STATE\_OPAQUE (1)
- #define OPTIX\_OPACITY\_MICROMAP\_STATE\_UNKNOWN\_TRANSPARENT (2)
- #define OPTIX\_OPACITY\_MICROMAP\_STATE\_UNKNOWN\_OPAQUE (3)
- #define OPTIX\_OPACITY\_MICROMAP\_PREDEFINED\_INDEX\_FULLY\_TRANSPARENT (-1)
- #define OPTIX\_OPACITY\_MICROMAP\_PREDEFINED\_INDEX\_FULLY\_OPAQUE (-2)
- #define OPTIX\_OPACITY\_MICROMAP\_PREDEFINED\_INDEX\_FULLY\_UNKNOWN\_TRANSPARENT (-3)
- #define OPTIX\_OPACITY\_MICROMAP\_PREDEFINED\_INDEX\_FULLY\_UNKNOWN\_OPAQUE (-4)
- #define OPTIX\_OPACITY\_MICROMAP\_ARRAY\_BUFFER\_BYTE\_ALIGNMENT 128ull
- #define OPTIX\_OPACITY\_MICROMAP\_MAX\_SUBDIVISION\_LEVEL 12
- #define OPTIX\_DISPLACEMENT\_MICROMAP\_MAX\_SUBDIVISION\_LEVEL 5
- #define OPTIX\_DISPLACEMENT\_MICROMAP\_DESC\_BUFFER\_BYTE\_ALIGNMENT 8ull
- #define OPTIX\_DISPLACEMENT\_MICROMAP\_ARRAY\_BUFFER\_BYTE\_ALIGNMENT 128ull

## Typedefs

- typedef unsigned long long CUdeviceptr
- typedef struct OptixDeviceContext\_t \* OptixDeviceContext
- typedef struct OptixModule\_t \* OptixModule
- typedef struct OptixProgramGroup\_t \* OptixProgramGroup
- typedef struct OptixPipeline\_t \* OptixPipeline
- typedef struct OptixDenoiser\_t \* OptixDenoiser
- typedef struct OptixTask\_t \* OptixTask
- typedef unsigned long long OptixTraversableHandle
- typedef unsigned int OptixVisibilityMask

- typedef enum OptixResult OptixResult
- typedef enum OptixDeviceProperty OptixDeviceProperty
- typedef void(\* OptixLogCallback) (unsigned int level, const char \*tag, const char \*message, void \*cbdata)
- typedef enum OptixDeviceContextValidationMode OptixDeviceContextValidationMode
- typedef struct OptixDeviceContextOptions OptixDeviceContextOptions
- typedef enum OptixGeometryFlags OptixGeometryFlags
- typedef enum OptixHitKind OptixHitKind
- typedef enum OptixIndicesFormat OptixIndicesFormat
- typedef enum OptixVertexFormat OptixVertexFormat
- typedef enum OptixTransformFormat OptixTransformFormat
- typedef enum OptixDisplacementMicromapBiasAndScaleFormat OptixDisplacementMicromapBiasAndScaleFormat
- typedef enum OptixDisplacementMicromapDirectionFormat OptixDisplacementMicromapDirectionFormat
- typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat
- typedef enum OptixOpacityMicromapArrayIndexingMode OptixOpacityMicromapArrayIndexingMode
- typedef struct OptixOpacityMicromapUsageCount OptixOpacityMicromapUsageCount
- typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap
- typedef struct OptixRelocateInputOpacityMicromap OptixRelocateInputOpacityMicromap
- typedef enum OptixDisplacementMicromapFormat OptixDisplacementMicromapFormat
- typedef enum OptixDisplacementMicromapFlags OptixDisplacementMicromapFlags
- typedef enum OptixDisplacementMicromapTriangleFlags OptixDisplacementMicromapTriangleFlags
- typedef struct OptixDisplacementMicromapDesc OptixDisplacementMicromapDesc
- typedef struct OptixDisplacementMicromapHistogramEntry OptixDisplacementMicromapHistogramEntry
- typedef struct OptixDisplacementMicromapArrayBuildInput OptixDisplacementMicromapArrayBuildInput
- typedef struct OptixDisplacementMicromapUsageCount OptixDisplacementMicromapUsageCount
- typedef enum OptixDisplacementMicromapArrayIndexingMode OptixDisplacementMicromapArrayIndexingMode
- typedef struct OptixBuildInputDisplacementMicromap OptixBuildInputDisplacementMicromap
- typedef struct OptixBuildInputTriangleArray OptixBuildInputTriangleArray
- typedef struct OptixRelocateInputTriangleArray OptixRelocateInputTriangleArray
- typedef enum OptixPrimitiveType OptixPrimitiveType
- typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags
- typedef enum OptixCurveEndcapFlags OptixCurveEndcapFlags
- typedef struct OptixBuildInputCurveArray OptixBuildInputCurveArray
- typedef struct OptixBuildInputSphereArray OptixBuildInputSphereArray
- typedef struct OptixAabb OptixAabb
- typedef struct OptixBuildInputCustomPrimitiveArray OptixBuildInputCustomPrimitiveArray
- typedef struct OptixBuildInputInstanceArray OptixBuildInputInstanceArray
- typedef struct OptixRelocateInputInstanceArray OptixRelocateInputInstanceArray
- typedef enum OptixBuildInputType OptixBuildInputType
- typedef struct OptixBuildInput OptixBuildInput
- typedef struct OptixRelocateInput OptixRelocateInput
- typedef enum OptixInstanceFlags OptixInstanceFlags



- typedef struct OptixInstance OptixInstance
- typedef enum OptixBuildFlags OptixBuildFlags
- typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags
- typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc
- typedef struct OptixOpacityMicromapHistogramEntry OptixOpacityMicromapHistogramEntry
- typedef struct OptixOpacityMicromapArrayBuildInput OptixOpacityMicromapArrayBuildInput
- typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
- typedef struct OptixMicromapBuffers OptixMicromapBuffers
- typedef enum OptixBuildOperation OptixBuildOperation
- typedef enum OptixMotionFlags OptixMotionFlags
- typedef struct OptixMotionOptions OptixMotionOptions
- typedef struct OptixAccelBuildOptions OptixAccelBuildOptions
- typedef struct OptixAccelBufferSizes OptixAccelBufferSizes
- typedef enum OptixAccelPropertyType OptixAccelPropertyType
- typedef struct OptixAccelEmitDesc OptixAccelEmitDesc
- typedef struct OptixRelocationInfo OptixRelocationInfo
- typedef struct OptixStaticTransform OptixStaticTransform
- typedef struct OptixMatrixMotionTransform OptixMatrixMotionTransform
- typedef struct OptixSRTData OptixSRTData
- typedef struct OptixSRTMotionTransform OptixSRTMotionTransform
- typedef enum OptixTraversableType OptixTraversableType
- typedef enum OptixPixelFormat OptixPixelFormat
- typedef struct OptixImage2D OptixImage2D
- typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
- typedef struct OptixDenoiserOptions OptixDenoiserOptions
- typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
- typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
- typedef struct OptixDenoiserLayer OptixDenoiserLayer
- typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
- typedef struct OptixDenoiserParams OptixDenoiserParams
- typedef struct OptixDenoiserSizes OptixDenoiserSizes
- typedef enum OptixRayFlags OptixRayFlags
- typedef enum OptixTransformType OptixTransformType
- typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags
- typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
- typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
- typedef enum OptixModuleCompileState OptixModuleCompileState
- typedef struct OptixModuleCompileBoundValueEntry OptixModuleCompileBoundValueEntry
- typedef enum OptixPayloadTypeID OptixPayloadTypeID
- typedef enum OptixPayloadSemantics OptixPayloadSemantics
- typedef struct OptixPayloadType OptixPayloadType
- typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
- typedef enum OptixProgramGroupKind OptixProgramGroupKind
- typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
- typedef struct OptixProgramGroupSingleModule OptixProgramGroupSingleModule
- typedef struct OptixProgramGroupHitgroup OptixProgramGroupHitgroup
- typedef struct OptixProgramGroupCallables OptixProgramGroupCallables
- typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
- typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
- typedef enum OptixExceptionCodes OptixExceptionCodes

- typedef enum OptixExceptionFlags OptixExceptionFlags
- typedef struct OptixPipelineCompileOptions OptixPipelineCompileOptions
- typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
- typedef struct OptixShaderBindingTable OptixShaderBindingTable
- typedef struct OptixStackSizes OptixStackSizes
- typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
- typedef OptixResult() OptixQueryFunctionTable\_t(int abiId, unsigned int numOptions, OptixQueryFunctionTableOptions \*, const void \*\*, void \*functionTable, size\_t sizeOfTable)
- typedef struct OptixBuiltinISOOptions OptixBuiltinISOOptions

## Enumerations

- enum OptixResult {  
OPTIX\_SUCCESS = 0 ,  
OPTIX\_ERROR\_INVALID\_VALUE = 7001 ,  
OPTIX\_ERROR\_HOST\_OUT\_OF\_MEMORY = 7002 ,  
OPTIX\_ERROR\_INVALID\_OPERATION = 7003 ,  
OPTIX\_ERROR\_FILE\_IO\_ERROR = 7004 ,  
OPTIX\_ERROR\_INVALID\_FILE\_FORMAT = 7005 ,  
OPTIX\_ERROR\_DISK\_CACHE\_INVALID\_PATH = 7010 ,  
OPTIX\_ERROR\_DISK\_CACHE\_PERMISSION\_ERROR = 7011 ,  
OPTIX\_ERROR\_DISK\_CACHE\_DATABASE\_ERROR = 7012 ,  
OPTIX\_ERROR\_DISK\_CACHE\_INVALID\_DATA = 7013 ,  
OPTIX\_ERROR\_LAUNCH\_FAILURE = 7050 ,  
OPTIX\_ERROR\_INVALID\_DEVICE\_CONTEXT = 7051 ,  
OPTIX\_ERROR\_CUDA\_NOT\_INITIALIZED = 7052 ,  
OPTIX\_ERROR\_VALIDATION\_FAILURE = 7053 ,  
OPTIX\_ERROR\_INVALID\_INPUT = 7200 ,  
OPTIX\_ERROR\_INVALID\_LAUNCH\_PARAMETER = 7201 ,  
OPTIX\_ERROR\_INVALID\_PAYLOAD\_ACCESS = 7202 ,  
OPTIX\_ERROR\_INVALID\_ATTRIBUTE\_ACCESS = 7203 ,  
OPTIX\_ERROR\_INVALID\_FUNCTION\_USE = 7204 ,  
OPTIX\_ERROR\_INVALID\_FUNCTION\_ARGUMENTS = 7205 ,  
OPTIX\_ERROR\_PIPELINE\_OUT\_OF\_CONSTANT\_MEMORY = 7250 ,  
OPTIX\_ERROR\_PIPELINE\_LINK\_ERROR = 7251 ,  
OPTIX\_ERROR\_ILLEGAL\_DURING\_TASK\_EXECUTE = 7270 ,  
OPTIX\_ERROR\_INTERNAL\_COMPILER\_ERROR = 7299 ,  
OPTIX\_ERROR\_DENOISER\_MODEL\_NOT\_SET = 7300 ,  
OPTIX\_ERROR\_DENOISER\_NOT\_INITIALIZED = 7301 ,  
OPTIX\_ERROR\_NOT\_COMPATIBLE = 7400 ,  
OPTIX\_ERROR\_PAYLOAD\_TYPE\_MISMATCH = 7500 ,  
OPTIX\_ERROR\_PAYLOAD\_TYPE\_RESOLUTION\_FAILED = 7501 ,  
OPTIX\_ERROR\_PAYLOAD\_TYPE\_ID\_INVALID = 7502 ,  
OPTIX\_ERROR\_NOT\_SUPPORTED = 7800 ,  
OPTIX\_ERROR\_UNSUPPORTED\_ABI\_VERSION = 7801 ,  
OPTIX\_ERROR\_FUNCTION\_TABLE\_SIZE\_MISMATCH = 7802 ,  
OPTIX\_ERROR\_INVALID\_ENTRY\_FUNCTION\_OPTIONS = 7803 ,  
OPTIX\_ERROR\_LIBRARY\_NOT\_FOUND = 7804 ,  
OPTIX\_ERROR\_ENTRY\_SYMBOL\_NOT\_FOUND = 7805 ,  
OPTIX\_ERROR\_LIBRARY\_UNLOAD\_FAILURE = 7806 ,  
OPTIX\_ERROR\_DEVICE\_OUT\_OF\_MEMORY = 7807 ,  
OPTIX\_ERROR\_CUDA\_ERROR = 7900 ,  
OPTIX\_ERROR\_INTERNAL\_ERROR = 7990 ,  
OPTIX\_ERROR\_UNKNOWN = 7999 }



- `enum OptixDeviceProperty {`  
`OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001 ,`  
`OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002 ,`  
`OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003 ,`  
`OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004 ,`  
`OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005 ,`  
`OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006 ,`  
`OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007 ,`  
`OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008 ,`  
`OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009 }`
- `enum OptixDeviceContextValidationMode {`  
`OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0 ,`  
`OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF }`
- `enum OptixGeometryFlags {`  
`OPTIX_GEOMETRY_FLAG_NONE = 0 ,`  
`OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u << 0 ,`  
`OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u << 1 ,`  
`OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 2 }`
- `enum OptixHitKind {`  
`OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE ,`  
`OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF }`
- `enum OptixIndicesFormat {`  
`OPTIX_INDICES_FORMAT_NONE = 0 ,`  
`OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102 ,`  
`OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103 }`
- `enum OptixVertexFormat {`  
`OPTIX_VERTEX_FORMAT_NONE = 0 ,`  
`OPTIX_VERTEX_FORMAT_FLOAT3 = 0x2121 ,`  
`OPTIX_VERTEX_FORMAT_FLOAT2 = 0x2122 ,`  
`OPTIX_VERTEX_FORMAT_HALF3 = 0x2123 ,`  
`OPTIX_VERTEX_FORMAT_HALF2 = 0x2124 ,`  
`OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125 ,`  
`OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126 }`
- `enum OptixTransformFormat {`  
`OPTIX_TRANSFORM_FORMAT_NONE = 0 ,`  
`OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1 }`
- `enum OptixDisplacementMicromapBiasAndScaleFormat {`  
`OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_NONE = 0 ,`  
`OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_FLOAT2 = 0x2241 ,`  
`OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_HALF2 = 0x2242 }`
- `enum OptixDisplacementMicromapDirectionFormat {`  
`OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_NONE = 0 ,`  
`OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_FLOAT3 = 0x2261 ,`  
`OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_HALF3 = 0x2262 }`
- `enum OptixOpacityMicromapFormat {`  
`OPTIX_OPACITY_MICROMAP_FORMAT_NONE = 0 ,`  
`OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE = 1 ,`  
`OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE = 2 }`
- `enum OptixOpacityMicromapArrayIndexingMode {`  
`OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0 ,`  
`OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1 ,`  
`OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2 }`

- enum OptixDisplacementMicromapFormat {  
OPTIX\_DISPLACEMENT\_MICROMAP\_FORMAT\_NONE = 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_FORMAT\_64\_MICRO\_TRIS\_64\_BYTES = 1 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_FORMAT\_256\_MICRO\_TRIS\_128\_BYTES = 2 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_FORMAT\_1024\_MICRO\_TRIS\_128\_BYTES = 3 }
- enum OptixDisplacementMicromapFlags {  
OPTIX\_DISPLACEMENT\_MICROMAP\_FLAG\_NONE = 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_FLAG\_PREFER\_FAST\_TRACE = 1 << 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_FLAG\_PREFER\_FAST\_BUILD = 1 << 1 }
- enum OptixDisplacementMicromapTriangleFlags {  
OPTIX\_DISPLACEMENT\_MICROMAP\_TRIANGLE\_FLAG\_NONE = 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_TRIANGLE\_FLAG\_DECIMATE\_EDGE\_01 = 1 << 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_TRIANGLE\_FLAG\_DECIMATE\_EDGE\_12 = 1 << 1 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_TRIANGLE\_FLAG\_DECIMATE\_EDGE\_20 = 1 << 2 }
- enum OptixDisplacementMicromapArrayIndexingMode {  
OPTIX\_DISPLACEMENT\_MICROMAP\_ARRAY\_INDEXING\_MODE\_NONE = 0 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_ARRAY\_INDEXING\_MODE\_LINEAR = 1 ,  
OPTIX\_DISPLACEMENT\_MICROMAP\_ARRAY\_INDEXING\_MODE\_INDEXED = 2 }
- enum OptixPrimitiveType {  
OPTIX\_PRIMITIVE\_TYPE\_CUSTOM = 0x2500 ,  
OPTIX\_PRIMITIVE\_TYPE\_ROUND\_QUADRATIC\_BSPLINE = 0x2501 ,  
OPTIX\_PRIMITIVE\_TYPE\_ROUND\_CUBIC\_BSPLINE = 0x2502 ,  
OPTIX\_PRIMITIVE\_TYPE\_ROUND\_LINEAR = 0x2503 ,  
OPTIX\_PRIMITIVE\_TYPE\_ROUND\_CATMULLROM = 0x2504 ,  
OPTIX\_PRIMITIVE\_TYPE\_FLAT\_QUADRATIC\_BSPLINE = 0x2505 ,  
OPTIX\_PRIMITIVE\_TYPE\_SPHERE = 0x2506 ,  
OPTIX\_PRIMITIVE\_TYPE\_ROUND\_CUBIC\_BEZIER = 0x2507 ,  
OPTIX\_PRIMITIVE\_TYPE\_TRIANGLE = 0x2531 ,  
OPTIX\_PRIMITIVE\_TYPE\_DISPLACED\_MICROMESH\_TRIANGLE = 0x2532 }
- enum OptixPrimitiveTypeFlags {  
OPTIX\_PRIMITIVE\_TYPE\_FLAGS\_CUSTOM = 1 << 0 ,  
OPTIX\_PRIMITIVE\_TYPE\_FLAGS\_ROUND\_QUADRATIC\_BSPLINE = 1 << 1 ,  
OPTIX\_PRIMITIVE\_TYPE\_FLAGS\_ROUND\_CUBIC\_BSPLINE = 1 << 2 ,  
OPTIX\_PRIMITIVE\_TYPE\_FLAGS\_ROUND\_LINEAR = 1 << 3 ,  
OPTIX\_PRIMITIVE\_TYPE\_FLAGS\_ROUND\_CATMULLROM = 1 << 4 ,  
OPTIX\_PRIMITIVE\_TYPE\_FLAGS\_FLAT\_QUADRATIC\_BSPLINE = 1 << 5 ,  
OPTIX\_PRIMITIVE\_TYPE\_FLAGS\_SPHERE = 1 << 6 ,  
OPTIX\_PRIMITIVE\_TYPE\_FLAGS\_ROUND\_CUBIC\_BEZIER = 1 << 7 ,  
OPTIX\_PRIMITIVE\_TYPE\_FLAGS\_TRIANGLE = 1 << 31 ,  
OPTIX\_PRIMITIVE\_TYPE\_FLAGS\_DISPLACED\_MICROMESH\_TRIANGLE = 1 << 30 }
- enum OptixCurveEndcapFlags {  
OPTIX\_CURVE\_ENDCAP\_DEFAULT = 0 ,  
OPTIX\_CURVE\_ENDCAP\_ON = 1 << 0 }
- enum OptixBuildInputType {  
OPTIX\_BUILD\_INPUT\_TYPE\_TRIANGLES = 0x2141 ,  
OPTIX\_BUILD\_INPUT\_TYPE\_CUSTOM\_PRIMITIVES = 0x2142 ,  
OPTIX\_BUILD\_INPUT\_TYPE\_INSTANCES = 0x2143 ,  
OPTIX\_BUILD\_INPUT\_TYPE\_INSTANCE\_POINTERS = 0x2144 ,  
OPTIX\_BUILD\_INPUT\_TYPE\_CURVES = 0x2145 ,  
OPTIX\_BUILD\_INPUT\_TYPE\_SPHERES = 0x2146 }
- enum OptixInstanceFlags {  
OPTIX\_INSTANCE\_FLAG\_NONE = 0 ,  
OPTIX\_INSTANCE\_FLAG\_DISABLE\_TRIANGLE\_FACE\_CULLING = 1u << 0 ,

```

OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u << 1,
OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT = 1u << 2,
OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u << 3,
OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 4,
OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS = 1u << 5 }
• enum OptixBuildFlags {
    OPTIX_BUILD_FLAG_NONE = 0,
    OPTIX_BUILD_FLAG_ALLOW_UPDATE = 1u << 0,
    OPTIX_BUILD_FLAG_ALLOW_COMPACTION = 1u << 1,
    OPTIX_BUILD_FLAG_PREFER_FAST_TRACE = 1u << 2,
    OPTIX_BUILD_FLAG_PREFER_FAST_BUILD = 1u << 3,
    OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS = 1u << 4,
    OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS = 1u << 5,
    OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE = 1u << 6,
    OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1u << 7 }
• enum OptixOpacityMicromapFlags {
    OPTIX_OPACITY_MICROMAP_FLAG_NONE = 0,
    OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 << 0,
    OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 << 1 }
• enum OptixBuildOperation {
    OPTIX_BUILD_OPERATION_BUILD = 0x2161,
    OPTIX_BUILD_OPERATION_UPDATE = 0x2162 }
• enum OptixMotionFlags {
    OPTIX_MOTION_FLAG_NONE = 0,
    OPTIX_MOTION_FLAG_START_VANISH = 1u << 0,
    OPTIX_MOTION_FLAG_END_VANISH = 1u << 1 }
• enum OptixAccelPropertyType {
    OPTIX_PROPERTY_TYPE_COMPACTED_SIZE = 0x2181,
    OPTIX_PROPERTY_TYPE_AABBS = 0x2182 }
• enum OptixTraversableType {
    OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM = 0x21C1,
    OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM = 0x21C2,
    OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM = 0x21C3 }
• enum OptixPixelFormat {
    OPTIX_PIXEL_FORMAT_HALF1 = 0x220a,
    OPTIX_PIXEL_FORMAT_HALF2 = 0x2207,
    OPTIX_PIXEL_FORMAT_HALF3 = 0x2201,
    OPTIX_PIXEL_FORMAT_HALF4 = 0x2202,
    OPTIX_PIXEL_FORMAT_FLOAT1 = 0x220b,
    OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208,
    OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203,
    OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204,
    OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205,
    OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206,
    OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209 }
• enum OptixDenoiserModelKind {
    OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322,
    OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323,
    OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324,
    OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325,
    OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326,
    OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327,
    OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328 }

```

- enum OptixDenoiserAOVType {  
OPTIX\_DENOISER\_AOV\_TYPE\_NONE = 0 ,  
OPTIX\_DENOISER\_AOV\_TYPE\_BEAUTY = 0x7000 ,  
OPTIX\_DENOISER\_AOV\_TYPE\_SPECULAR = 0x7001 ,  
OPTIX\_DENOISER\_AOV\_TYPE\_REFLECTION = 0x7002 ,  
OPTIX\_DENOISER\_AOV\_TYPE\_REFRACTION = 0x7003 ,  
OPTIX\_DENOISER\_AOV\_TYPE\_DIFFUSE = 0x7004 }
- enum OptixDenoiserAlphaMode {  
OPTIX\_DENOISER\_ALPHA\_MODE\_COPY = 0 ,  
OPTIX\_DENOISER\_ALPHA\_MODE\_ALPHA\_AS\_AOV = 1 ,  
OPTIX\_DENOISER\_ALPHA\_MODE\_FULL\_DENOISE\_PASS = 2 }
- enum OptixRayFlags {  
OPTIX\_RAY\_FLAG\_NONE = 0u ,  
OPTIX\_RAY\_FLAG\_DISABLE\_ANYHIT = 1u << 0 ,  
OPTIX\_RAY\_FLAG\_ENFORCE\_ANYHIT = 1u << 1 ,  
OPTIX\_RAY\_FLAG\_TERMINATE\_ON\_FIRST\_HIT = 1u << 2 ,  
OPTIX\_RAY\_FLAG\_DISABLE\_CLOSESTHIT = 1u << 3 ,  
OPTIX\_RAY\_FLAG\_CULL\_BACK\_FACING\_TRIANGLES = 1u << 4 ,  
OPTIX\_RAY\_FLAG\_CULL\_FRONT\_FACING\_TRIANGLES = 1u << 5 ,  
OPTIX\_RAY\_FLAG\_CULL\_DISABLED\_ANYHIT = 1u << 6 ,  
OPTIX\_RAY\_FLAG\_CULL\_ENFORCED\_ANYHIT = 1u << 7 ,  
OPTIX\_RAY\_FLAG\_FORCE\_OPACITY\_MICROMAP\_2\_STATE = 1u << 10 }
- enum OptixTransformType {  
OPTIX\_TRANSFORM\_TYPE\_NONE = 0 ,  
OPTIX\_TRANSFORM\_TYPE\_STATIC\_TRANSFORM = 1 ,  
OPTIX\_TRANSFORM\_TYPE\_MATRIX\_MOTION\_TRANSFORM = 2 ,  
OPTIX\_TRANSFORM\_TYPE\_SRT\_MOTION\_TRANSFORM = 3 ,  
OPTIX\_TRANSFORM\_TYPE\_INSTANCE = 4 }
- enum OptixTraversableGraphFlags {  
OPTIX\_TRAVERSABLE\_GRAPH\_FLAG\_ALLOW\_ANY = 0 ,  
OPTIX\_TRAVERSABLE\_GRAPH\_FLAG\_ALLOW\_SINGLE\_GAS = 1u << 0 ,  
OPTIX\_TRAVERSABLE\_GRAPH\_FLAG\_ALLOW\_SINGLE\_LEVEL\_INSTANCING = 1u << 1 }
- enum OptixCompileOptimizationLevel {  
OPTIX\_COMPILE\_OPTIMIZATION\_DEFAULT = 0 ,  
OPTIX\_COMPILE\_OPTIMIZATION\_LEVEL\_0 = 0x2340 ,  
OPTIX\_COMPILE\_OPTIMIZATION\_LEVEL\_1 = 0x2341 ,  
OPTIX\_COMPILE\_OPTIMIZATION\_LEVEL\_2 = 0x2342 ,  
OPTIX\_COMPILE\_OPTIMIZATION\_LEVEL\_3 = 0x2343 }
- enum OptixCompileDebugLevel {  
OPTIX\_COMPILE\_DEBUG\_LEVEL\_DEFAULT = 0 ,  
OPTIX\_COMPILE\_DEBUG\_LEVEL\_NONE = 0x2350 ,  
OPTIX\_COMPILE\_DEBUG\_LEVEL\_MINIMAL = 0x2351 ,  
OPTIX\_COMPILE\_DEBUG\_LEVEL\_MODERATE = 0x2353 ,  
OPTIX\_COMPILE\_DEBUG\_LEVEL\_FULL = 0x2352 }
- enum OptixModuleCompileState {  
OPTIX\_MODULE\_COMPILE\_STATE\_NOT\_STARTED = 0x2360 ,  
OPTIX\_MODULE\_COMPILE\_STATE\_STARTED = 0x2361 ,  
OPTIX\_MODULE\_COMPILE\_STATE\_IMPENDING\_FAILURE = 0x2362 ,  
OPTIX\_MODULE\_COMPILE\_STATE\_FAILED = 0x2363 ,  
OPTIX\_MODULE\_COMPILE\_STATE\_COMPLETED = 0x2364 }
- enum OptixPayloadTypeID {  
OPTIX\_PAYLOAD\_TYPE\_DEFAULT = 0 ,  
OPTIX\_PAYLOAD\_TYPE\_ID\_0 = (1 << 0u) ,

```

OPTIX_PAYLOAD_TYPE_ID_1 = (1 << 1u) ,
OPTIX_PAYLOAD_TYPE_ID_2 = (1 << 2u) ,
OPTIX_PAYLOAD_TYPE_ID_3 = (1 << 3u) ,
OPTIX_PAYLOAD_TYPE_ID_4 = (1 << 4u) ,
OPTIX_PAYLOAD_TYPE_ID_5 = (1 << 5u) ,
OPTIX_PAYLOAD_TYPE_ID_6 = (1 << 6u) ,
OPTIX_PAYLOAD_TYPE_ID_7 = (1 << 7u) }
• enum OptixPayloadSemantics {
    OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE = 0 ,
    OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ = 1u << 0 ,
    OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE = 2u << 0 ,
    OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u << 0 ,
    OPTIX_PAYLOAD_SEMANTICS_CH_NONE = 0 ,
    OPTIX_PAYLOAD_SEMANTICS_CH_READ = 1u << 2 ,
    OPTIX_PAYLOAD_SEMANTICS_CH_WRITE = 2u << 2 ,
    OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE = 3u << 2 ,
    OPTIX_PAYLOAD_SEMANTICS_MS_NONE = 0 ,
    OPTIX_PAYLOAD_SEMANTICS_MS_READ = 1u << 4 ,
    OPTIX_PAYLOAD_SEMANTICS_MS_WRITE = 2u << 4 ,
    OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE = 3u << 4 ,
    OPTIX_PAYLOAD_SEMANTICS_AH_NONE = 0 ,
    OPTIX_PAYLOAD_SEMANTICS_AH_READ = 1u << 6 ,
    OPTIX_PAYLOAD_SEMANTICS_AH_WRITE = 2u << 6 ,
    OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE = 3u << 6 ,
    OPTIX_PAYLOAD_SEMANTICS_IS_NONE = 0 ,
    OPTIX_PAYLOAD_SEMANTICS_IS_READ = 1u << 8 ,
    OPTIX_PAYLOAD_SEMANTICS_IS_WRITE = 2u << 8 ,
    OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE = 3u << 8 }
• enum OptixProgramGroupKind {
    OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421 ,
    OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422 ,
    OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423 ,
    OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424 ,
    OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425 }
• enum OptixProgramGroupFlags { OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0 }
• enum OptixExceptionCodes {
    OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1 ,
    OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2 ,
    OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED = -3 ,
    OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE = -5 ,
    OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT = -6 ,
    OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT = -7 ,
    OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE = -8 ,
    OPTIX_EXCEPTION_CODE_INVALID_RAY = -9 ,
    OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH = -10 ,
    OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH = -11 ,
    OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT = -12 ,
    OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD = -13 ,
    OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD = -14 ,
    OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS = -15 ,
    OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0 = -16 ,
    OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_1 = -17 ,
    OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2 = -18 ,

```

```

    OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS = -32 ,
    OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH = -33 }
• enum OptixExceptionFlags {
    OPTIX_EXCEPTION_FLAG_NONE = 0 ,
    OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u << 0 ,
    OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u << 1 ,
    OPTIX_EXCEPTION_FLAG_USER = 1u << 2 ,
    OPTIX_EXCEPTION_FLAG_DEBUG = 1u << 3 }
• enum OptixQueryFunctionTableOptions { OPTIX_QUERY_FUNCTION_TABLE_OPTION_
    DUMMY = 0 }

```

### 8.23.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

OptiX types include file – defines types and enums used by the API. For the math library routines include `optix_math.h`

## 8.24 optix\_types.h

[Go to the documentation of this file.](#)

```

1
2 /*
3 * Copyright (c) 2021 NVIDIA Corporation. All rights reserved.
4 *
5 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
6 * rights in and to this software, related documentation and any modifications thereto.
7 * Any use, reproduction, disclosure or distribution of this software and related
8 * documentation without an express license agreement from NVIDIA Corporation is strictly
9 * prohibited.
10 *
11 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
12 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
13 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
14 * PARTICULAR PURPOSE. IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
15 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
16 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
17 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
18 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
19 * SUCH DAMAGES
20 */
21
22
23
24
25
26
27
28
29 #ifndef OPTIX_OPTIX_TYPES_H
30 #define OPTIX_OPTIX_TYPES_H
31
32 #if !defined(__CUDACC_RTC__)
33 #include <stddef.h> /* for size_t */
34 #endif
35
36 #ifdef NV_MODULE_OPTIX
37 // This is a mechanism to include <g_nvconfig.h> in driver builds only and translate any nvconfig macro to
38 // a custom OPTIX-specific macro, that can also be used in SDK builds/installs
39 #include <exp/misc/optix_nvconfig_translate.h> // includes <g_nvconfig.h>
40 #endif // NV_MODULE_OPTIX
41
42
43
44

```



```

49 // This typedef should match the one in cuda.h in order to avoid compilation errors.
50 #if defined(_WIN64) || defined(__LP64__)
52 typedef unsigned long long CUdeviceptr;
53 #else
55 typedef unsigned int CUdeviceptr;
56 #endif
57
59 typedef struct OptixDeviceContext_t* OptixDeviceContext;
60
62 typedef struct OptixModule_t* OptixModule;
63
65 typedef struct OptixProgramGroup_t* OptixProgramGroup;
66
68 typedef struct OptixPipeline_t* OptixPipeline;
69
71 typedef struct OptixDenoiser_t* OptixDenoiser;
72
74 typedef struct OptixTask_t* OptixTask;
75
77 typedef unsigned long long OptixTraversableHandle;
78
80 typedef unsigned int OptixVisibilityMask;
81
83 #define OPTIX_SBT_RECORD_HEADER_SIZE ((size_t)32)
84
86 #define OPTIX_SBT_RECORD_ALIGNMENT 16ull
87
89 #define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128ull
90
92 #define OPTIX_INSTANCE_BYTE_ALIGNMENT 16ull
93
95 #define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8ull
96
98 #define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16ull
99
101 #define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
102
104 #define OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
105
107 #define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
108
110 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
111
113 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
114
117 #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT (0)
118 #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE (1)
119 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT (2)
120 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE (3)
121
124 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT (-1)
125 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE (-2)
126 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT (-3)
127 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE (-4)
128
130 #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
131
133 #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
134
136 #define OPTIX_DISPLACEMENT_MICROMAP_MAX_SUBDIVISION_LEVEL 5
137
139 #define OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
140
142 #define OPTIX_DISPLACEMENT_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
143
151 typedef enum OptixResult
152 {

```

```

153     OPTIX_SUCCESS = 0,
154     OPTIX_ERROR_INVALID_VALUE = 7001,
155     OPTIX_ERROR_HOST_OUT_OF_MEMORY = 7002,
156     OPTIX_ERROR_INVALID_OPERATION = 7003,
157     OPTIX_ERROR_FILE_IO_ERROR = 7004,
158     OPTIX_ERROR_INVALID_FILE_FORMAT = 7005,
159     OPTIX_ERROR_DISK_CACHE_INVALID_PATH = 7010,
160     OPTIX_ERROR_DISK_CACHE_PERMISSION_ERROR = 7011,
161     OPTIX_ERROR_DISK_CACHE_DATABASE_ERROR = 7012,
162     OPTIX_ERROR_DISK_CACHE_INVALID_DATA = 7013,
163     OPTIX_ERROR_LAUNCH_FAILURE = 7050,
164     OPTIX_ERROR_INVALID_DEVICE_CONTEXT = 7051,
165     OPTIX_ERROR_CUDA_NOT_INITIALIZED = 7052,
166     OPTIX_ERROR_VALIDATION_FAILURE = 7053,
167     OPTIX_ERROR_INVALID_INPUT = 7200,
168     OPTIX_ERROR_INVALID_LAUNCH_PARAMETER = 7201,
169     OPTIX_ERROR_INVALID_PAYLOAD_ACCESS = 7202,
170     OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS = 7203,
171     OPTIX_ERROR_INVALID_FUNCTION_USE = 7204,
172     OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS = 7205,
173     OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250,
174     OPTIX_ERROR_PIPELINE_LINK_ERROR = 7251,
175     OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE = 7270,
176     OPTIX_ERROR_INTERNAL_COMPILER_ERROR = 7299,
177     OPTIX_ERROR_DENOISER_MODEL_NOT_SET = 7300,
178     OPTIX_ERROR_DENOISER_NOT_INITIALIZED = 7301,
179     OPTIX_ERROR_NOT_COMPATIBLE = 7400,
180     OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH = 7500,
181     OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED = 7501,
182     OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID = 7502,
183     OPTIX_ERROR_NOT_SUPPORTED = 7800,
184     OPTIX_ERROR_UNSUPPORTED_ABI_VERSION = 7801,
185     OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH = 7802,
186     OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS = 7803,
187     OPTIX_ERROR_LIBRARY_NOT_FOUND = 7804,
188     OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND = 7805,
189     OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE = 7806,
190     OPTIX_ERROR_DEVICE_OUT_OF_MEMORY = 7807,
191     OPTIX_ERROR_CUDA_ERROR = 7900,
192     OPTIX_ERROR_INTERNAL_ERROR = 7990,
193     OPTIX_ERROR_UNKNOWN = 7999,
194 } OptixResult;
195
196 typedef enum OptixDeviceProperty
197 {
198     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001,
199
200     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002,
201
202     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003,
203
204     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004,
205
206     OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005,
207
208     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006,
209
210     OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007,
211
212     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008,
213
214     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009,
215 } OptixDeviceProperty;
216
217 typedef void (*OptixLogCallback)(unsigned int level, const char* tag, const char* message, void* cbdata);
218
219 typedef enum OptixDeviceContextValidationMode

```



```

271 {
272     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0,
273     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF
274 } OptixDeviceContextValidationMode;
275
276 typedef struct OptixDeviceContextOptions
277 {
278     OptixLogCallback logCallbackFunction;
279     void* logCallbackData;
280     int logCallbackLevel;
281     OptixDeviceContextValidationMode validationMode;
282 } OptixDeviceContextOptions;
283
284 typedef enum OptixGeometryFlags
285 {
286     OPTIX_GEOMETRY_FLAG_NONE = 0,
287
288     OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u « 0,
289
290     OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u « 1,
291
292     OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u « 2,
293 } OptixGeometryFlags;
294
295 typedef enum OptixHitKind
296 {
297     OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE,
298     OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF
299 } OptixHitKind;
300
301 typedef enum OptixIndicesFormat
302 {
303     OPTIX_INDICES_FORMAT_NONE = 0,
304     OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102,
305     OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103
306 } OptixIndicesFormat;
307
308 typedef enum OptixVertexFormat
309 {
310     OPTIX_VERTEX_FORMAT_NONE = 0,
311     OPTIX_VERTEX_FORMAT_FLOAT3 = 0x2121,
312     OPTIX_VERTEX_FORMAT_FLOAT2 = 0x2122,
313     OPTIX_VERTEX_FORMAT_HALF3 = 0x2123,
314     OPTIX_VERTEX_FORMAT_HALF2 = 0x2124,
315     OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125,
316     OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126
317 } OptixVertexFormat;
318
319 typedef enum OptixTransformFormat
320 {
321     OPTIX_TRANSFORM_FORMAT_NONE = 0,
322     OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1,
323 } OptixTransformFormat;
324
325 typedef enum OptixDisplacementMicromapBiasAndScaleFormat
326 {
327     OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_NONE = 0,
328     OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_FLOAT2 = 0x2241,
329     OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_HALF2 = 0x2242,
330 } OptixDisplacementMicromapBiasAndScaleFormat;
331
332 typedef enum OptixDisplacementMicromapDirectionFormat
333 {
334     OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_NONE = 0,
335     OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_FLOAT3 = 0x2261,
336     OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_HALF3 = 0x2262,
337 } OptixDisplacementMicromapDirectionFormat;

```

```

369
371 typedef enum OptixOpacityMicromapFormat
372 {
374     OPTIX_OPACITY_MICROMAP_FORMAT_NONE = 0,
376     OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE = 1,
378     OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE = 2,
379 } OptixOpacityMicromapFormat;
380
382 typedef enum OptixOpacityMicromapArrayIndexingMode
383 {
385     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0,
388     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1,
392     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2,
393 } OptixOpacityMicromapArrayIndexingMode;
394
399 typedef struct OptixOpacityMicromapUsageCount
400 {
403     unsigned int count;
405     unsigned int subdivisionLevel;
407     OptixOpacityMicromapFormat format;
408 } OptixOpacityMicromapUsageCount;
409
410 typedef struct OptixBuildInputOpacityMicromap
411 {
413     OptixOpacityMicromapArrayIndexingMode indexingMode;
414
419     CUdeviceptr opacityMicromapArray;
420
430     CUdeviceptr indexBuffer;
431
434     unsigned int indexSizeInBytes;
435
438     unsigned int indexStrideInBytes;
439
441     unsigned int indexOffset;
442
444     unsigned int numMicromapUsageCounts;
447     const OptixOpacityMicromapUsageCount* micromapUsageCounts;
448 } OptixBuildInputOpacityMicromap;
449
450 typedef struct OptixRelocateInputOpacityMicromap
451 {
455     CUdeviceptr opacityMicromapArray;
456 } OptixRelocateInputOpacityMicromap;
457
458
460 typedef enum OptixDisplacementMicromapFormat
461 {
462     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_NONE = 0,
463     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_64_MICRO_TRIS_64_BYTES = 1,
464     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_256_MICRO_TRIS_128_BYTES = 2,
465     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_1024_MICRO_TRIS_128_BYTES = 3,
466 } OptixDisplacementMicromapFormat;
467
469 typedef enum OptixDisplacementMicromapFlags
470 {
471     OPTIX_DISPLACEMENT_MICROMAP_FLAG_NONE = 0,
472
474     OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 « 0,
475
477     OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 « 1,
478
479 } OptixDisplacementMicromapFlags;
480
481 typedef enum OptixDisplacementMicromapTriangleFlags
482 {
483     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_NONE = 0,

```

```

486     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_01 = 1 « 0,
488     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_12 = 1 « 1,
490     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_20 = 1 « 2,
491 } OptixDisplacementMicromapTriangleFlags;
492
493 typedef struct OptixDisplacementMicromapDesc
494 {
495     unsigned int    byteOffset;
496     unsigned short subdivisionLevel;
497     unsigned short format;
498 } OptixDisplacementMicromapDesc;
499
500 typedef struct OptixDisplacementMicromapHistogramEntry
501 {
502     unsigned int    count;
503     unsigned int    subdivisionLevel;
504     OptixDisplacementMicromapFormat format;
505 } OptixDisplacementMicromapHistogramEntry;
506
507 typedef struct OptixDisplacementMicromapArrayBuildInput
508 {
509     OptixDisplacementMicromapFlags    flags;
510     CUdeviceptr displacementValuesBuffer;
511     CUdeviceptr perDisplacementMicromapDescBuffer;
512     unsigned int perDisplacementMicromapDescStrideInBytes;
513     unsigned int numDisplacementMicromapHistogramEntries;
514     const OptixDisplacementMicromapHistogramEntry* displacementMicromapHistogramEntries;
515 } OptixDisplacementMicromapArrayBuildInput;
516
517 typedef struct OptixDisplacementMicromapUsageCount
518 {
519     unsigned int    count;
520     unsigned int    subdivisionLevel;
521     OptixDisplacementMicromapFormat format;
522 } OptixDisplacementMicromapUsageCount;
523
524 typedef enum OptixDisplacementMicromapArrayIndexingMode
525 {
526     OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0,
527     OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1,
528     OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2,
529 } OptixDisplacementMicromapArrayIndexingMode;
530
531 typedef struct OptixBuildInputDisplacementMicromap
532 {
533     OptixDisplacementMicromapArrayIndexingMode indexingMode;
534
535     CUdeviceptr displacementMicromapArray;
536     CUdeviceptr displacementMicromapIndexBuffer;
537     CUdeviceptr vertexDirectionsBuffer;
538     CUdeviceptr vertexBiasAndScaleBuffer;
539     CUdeviceptr triangleFlagsBuffer;
540
541     unsigned int displacementMicromapIndexOffset;
542     unsigned int displacementMicromapIndexStrideInBytes;
543     unsigned int displacementMicromapIndexSizeInBytes;
544
545     OptixDisplacementMicromapDirectionFormat vertexDirectionFormat;
546     unsigned int vertexDirectionStrideInBytes;
547
548     OptixDisplacementMicromapBiasAndScaleFormat vertexBiasAndScaleFormat;
549     unsigned int vertexBiasAndScaleStrideInBytes;
550
551     unsigned int triangleFlagsStrideInBytes;
552
553     unsigned int numDisplacementMicromapUsageCounts;
554     const OptixDisplacementMicromapUsageCount* displacementMicromapUsageCounts;

```

```

610
611 } OptixBuildInputDisplacementMicromap;
612
613
617 typedef struct OptixBuildInputTriangleArray
618 {
626     const CUdeviceptr* vertexBuffers;
627
629     unsigned int numVertices;
630
632     OptixVertexFormat vertexFormat;
633
636     unsigned int vertexStrideInBytes;
637
641     CUdeviceptr indexBuffer;
642
644     unsigned int numIndexTriplets;
645
647     OptixIndicesFormat indexFormat;
648
651     unsigned int indexStrideInBytes;
652
656     CUdeviceptr preTransform;
657
661     const unsigned int* flags;
662
664     unsigned int numSbtRecords;
665
669     CUdeviceptr sbtIndexOffsetBuffer;
670
672     unsigned int sbtIndexOffsetSizeInBytes;
673
676     unsigned int sbtIndexOffsetStrideInBytes;
677
680     unsigned int primitiveIndexOffset;
681
683     OptixTransformFormat transformFormat;
684
686     OptixBuildInputOpacityMicromap opacityMicromap;
688     OptixBuildInputDisplacementMicromap displacementMicromap;
689
690 } OptixBuildInputTriangleArray;
691
695 typedef struct OptixRelocateInputTriangleArray
696 {
699     unsigned int numSbtRecords;
700
702     OptixRelocateInputOpacityMicromap opacityMicromap;
703 } OptixRelocateInputTriangleArray;
704
707 typedef enum OptixPrimitiveType
708 {
710     OPTIX_PRIMITIVE_TYPE_CUSTOM = 0x2500,
712     OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE = 0x2501,
714     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE = 0x2502,
716     OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR = 0x2503,
718     OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM = 0x2504,
720     OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE = 0x2505,
722     OPTIX_PRIMITIVE_TYPE_SPHERE = 0x2506,
724     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER = 0x2507,
726     OPTIX_PRIMITIVE_TYPE_TRIANGLE = 0x2531,
728     OPTIX_PRIMITIVE_TYPE_DISPLACED_MICROMESH_TRIANGLE = 0x2532,
729 } OptixPrimitiveType;
730
734 typedef enum OptixPrimitiveTypeFlags
735 {
737     OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM = 1 « 0,

```

```

739     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE      = 1 « 1,
741     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE         = 1 « 2,
743     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR                 = 1 « 3,
745     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM             = 1 « 4,
747     OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE       = 1 « 5,
749     OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE                       = 1 « 6,
751     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER           = 1 « 7,
753     OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE                     = 1 « 31,
755     OPTIX_PRIMITIVE_TYPE_FLAGS_DISPLACED_MICROMESH_TRIANGLE = 1 « 30,
756 } OptixPrimitiveTypeFlags;
757
760 typedef enum OptixCurveEndcapFlags
761 {
763     OPTIX_CURVE_ENDCAP_DEFAULT      = 0,
765     OPTIX_CURVE_ENDCAP_ON           = 1 « 0,
766 } OptixCurveEndcapFlags;
767
785 typedef struct OptixBuildInputCurveArray
786 {
789     OptixPrimitiveType curveType;
791     unsigned int numPrimitives;
792
797     const CUdeviceptr* vertexBuffers;
799     unsigned int numVertices;
802     unsigned int vertexStrideInBytes;
803
806     const CUdeviceptr* widthBuffers;
809     unsigned int widthStrideInBytes;
810
812     const CUdeviceptr* normalBuffers;
814     unsigned int normalStrideInBytes;
815
821     CUdeviceptr indexBuffer;
824     unsigned int indexStrideInBytes;
825
828     unsigned int flag;
829
832     unsigned int primitiveIndexOffset;
833
835     unsigned int endcapFlags;
836 } OptixBuildInputCurveArray;
837
850 typedef struct OptixBuildInputSphereArray
851 {
856     const CUdeviceptr* vertexBuffers;
857
860     unsigned int vertexStrideInBytes;
862     unsigned int numVertices;
863
866     const CUdeviceptr* radiusBuffers;
869     unsigned int radiusStrideInBytes;
872     int singleRadius;
873
877     const unsigned int* flags;
878
880     unsigned int numSbtRecords;
884     CUdeviceptr sbtIndexOffsetBuffer;
886     unsigned int sbtIndexOffsetSizeInBytes;
889     unsigned int sbtIndexOffsetStrideInBytes;
890
893     unsigned int primitiveIndexOffset;
894 } OptixBuildInputSphereArray;
895
897 typedef struct OptixAabb
898 {
899     float minX;
900     float minY;

```

```

901     float minZ;
902     float maxX;
903     float maxY;
904     float maxZ;
905 } OptixAabb;
906
910 typedef struct OptixBuildInputCustomPrimitiveArray
911 {
916     const CUdeviceptr* aabbBuffers;
917
920     unsigned int numPrimitives;
921
925     unsigned int strideInBytes;
926
930     const unsigned int* flags;
931
933     unsigned int numSbtRecords;
934
938     CUdeviceptr sbtIndexOffsetBuffer;
939
941     unsigned int sbtIndexOffsetSizeInBytes;
942
945     unsigned int sbtIndexOffsetStrideInBytes;
946
949     unsigned int primitiveIndexOffset;
950 } OptixBuildInputCustomPrimitiveArray;
951
955 typedef struct OptixBuildInputInstanceArray
956 {
964     CUdeviceptr instances;
965
967     unsigned int numInstances;
968
972     unsigned int instanceStride;
973 } OptixBuildInputInstanceArray;
974
978 typedef struct OptixRelocateInputInstanceArray
979 {
982     unsigned int numInstances;
983
989     CUdeviceptr traversableHandles;
990
991 } OptixRelocateInputInstanceArray;
992
996 typedef enum OptixBuildInputType
997 {
999     OPTIX_BUILD_INPUT_TYPE_TRIANGLES = 0x2141,
1001     OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES = 0x2142,
1003     OPTIX_BUILD_INPUT_TYPE_INSTANCES = 0x2143,
1005     OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS = 0x2144,
1007     OPTIX_BUILD_INPUT_TYPE_CURVES = 0x2145,
1009     OPTIX_BUILD_INPUT_TYPE_SPHERES = 0x2146
1010 } OptixBuildInputType;
1011
1017 typedef struct OptixBuildInput
1018 {
1020     OptixBuildInputType type;
1021
1022     union
1023     {
1025         OptixBuildInputTriangleArray triangleArray;
1027         OptixBuildInputCurveArray curveArray;
1029         OptixBuildInputSphereArray sphereArray;
1031         OptixBuildInputCustomPrimitiveArray customPrimitiveArray;
1033         OptixBuildInputInstanceArray instanceArray;
1034         char pad[1024];
1035     };

```

```

1036 } OptixBuildInput;
1037
1041 typedef struct OptixRelocateInput
1042 {
1044     OptixBuildInputType type;
1045
1046     union
1047     {
1049         OptixRelocateInputInstanceArray instanceArray;
1050
1052         OptixRelocateInputTriangleArray triangleArray;
1053     };
1055 } OptixRelocateInput;
1056
1058 // Some 32-bit tools use this header. This static_assert fails for them because
1059 // the default enum size is 4 bytes, rather than 8, under 32-bit compilers.
1060 // This #ifndef allows them to disable the static assert.
1061
1062 // TODO Define a static assert for C/pre-C++-11
1063 #if defined(__cplusplus) && __cplusplus >= 201103L
1064 static_assert(sizeof(OptixBuildInput) == 8 + 1024, "OptixBuildInput has wrong size");
1065 #endif
1066
1070 typedef enum OptixInstanceFlags
1071 {
1073     OPTIX_INSTANCE_FLAG_NONE = 0,
1074
1078     OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 0,
1079
1082     OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u << 1,
1083
1087     OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT = 1u << 2,
1088
1093     OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u << 3,
1094
1095
1097     OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 4,
1100     OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS = 1u << 5,
1101
1102 } OptixInstanceFlags;
1103
1107 typedef struct OptixInstance
1108 {
1110     float transform[12];
1111
1113     unsigned int instanceId;
1114
1118     unsigned int sbtOffset;
1119
1122     unsigned int visibilityMask;
1123
1125     unsigned int flags;
1126
1128     OptixTraversableHandle traversableHandle;
1129
1131     unsigned int pad[2];
1132 } OptixInstance;
1133
1137 typedef enum OptixBuildFlags
1138 {
1140     OPTIX_BUILD_FLAG_NONE = 0,
1141
1144     OPTIX_BUILD_FLAG_ALLOW_UPDATE = 1u << 0,
1145
1146     OPTIX_BUILD_FLAG_ALLOW_COMPACTION = 1u << 1,
1147

```

```

1149     OPTIX_BUILD_FLAG_PREFER_FAST_TRACE = 1u « 2,
1150
1152     OPTIX_BUILD_FLAG_PREFER_FAST_BUILD = 1u « 3,
1153
1163     OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS = 1u « 4,
1164
1167     OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS = 1u « 5,
1168
1172     OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE = 1u « 6,
1173
1177     OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1u « 7,
1178 } OptixBuildFlags;
1179
1180
1182 typedef enum OptixOpacityMicromapFlags
1183 {
1184     OPTIX_OPACITY_MICROMAP_FLAG_NONE = 0,
1185
1187     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 « 0,
1188
1190     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 « 1,
1191 } OptixOpacityMicromapFlags;
1192
1194 typedef struct OptixOpacityMicromapDesc
1195 {
1197     unsigned int    byteOffset;
1199     unsigned short subdivisionLevel;
1201     unsigned short format;
1202 } OptixOpacityMicromapDesc;
1203
1208 typedef struct OptixOpacityMicromapHistogramEntry
1209 {
1211     unsigned int    count;
1213     unsigned int    subdivisionLevel;
1215     OptixOpacityMicromapFormat format;
1216 } OptixOpacityMicromapHistogramEntry;
1217
1219 typedef struct OptixOpacityMicromapArrayBuildInput
1220 {
1222     unsigned int flags;
1223
1225     CUdeviceptr inputBuffer;
1226
1229     CUdeviceptr perMicromapDescBuffer;
1230
1234     unsigned int perMicromapDescStrideInBytes;
1235
1237     unsigned int numMicromapHistogramEntries;
1240     const OptixOpacityMicromapHistogramEntry* micromapHistogramEntries;
1241 } OptixOpacityMicromapArrayBuildInput;
1242
1244 typedef struct OptixMicromapBufferSizes
1245 {
1246     size_t outputSizeInBytes;
1247     size_t tempSizeInBytes;
1248 } OptixMicromapBufferSizes;
1249
1251 typedef struct OptixMicromapBuffers
1252 {
1254     CUdeviceptr output;
1256     size_t outputSizeInBytes;
1258     CUdeviceptr temp;
1260     size_t tempSizeInBytes;
1261 } OptixMicromapBuffers;
1262
1263
1275 typedef enum OptixBuildOperation

```



```

1276 {
1277     OPTIX_BUILD_OPERATION_BUILD = 0x2161,
1278     OPTIX_BUILD_OPERATION_UPDATE = 0x2162,
1279 } OptixBuildOperation;
1280
1281 typedef enum OptixMotionFlags
1282 {
1283     OPTIX_MOTION_FLAG_NONE = 0,
1284     OPTIX_MOTION_FLAG_START_VANISH = 1u « 0,
1285     OPTIX_MOTION_FLAG_END_VANISH = 1u « 1
1286 } OptixMotionFlags;
1287
1288 typedef struct OptixMotionOptions
1289 {
1290     unsigned short numKeys;
1291     unsigned short flags;
1292     float timeBegin;
1293     float timeEnd;
1294 } OptixMotionOptions;
1295
1296 typedef struct OptixAccelBuildOptions
1297 {
1298     unsigned int buildFlags;
1299     OptixBuildOperation operation;
1300     OptixMotionOptions motionOptions;
1301 } OptixAccelBuildOptions;
1302
1303 typedef struct OptixAccelBufferSizes
1304 {
1305     size_t outputSizeInBytes;
1306     size_t tempSizeInBytes;
1307     size_t tempUpdateSizeInBytes;
1308 } OptixAccelBufferSizes;
1309
1310 typedef enum OptixAccelPropertyType
1311 {
1312     OPTIX_PROPERTY_TYPE_COMPACTED_SIZE = 0x2181,
1313     OPTIX_PROPERTY_TYPE_AABBS = 0x2182,
1314 } OptixAccelPropertyType;
1315
1316 typedef struct OptixAccelEmitDesc
1317 {
1318     CUdeviceptr result;
1319     OptixAccelPropertyType type;
1320 } OptixAccelEmitDesc;
1321
1322 typedef struct OptixRelocationInfo
1323 {
1324     unsigned long long info[4];
1325 } OptixRelocationInfo;
1326
1327 typedef struct OptixStaticTransform
1328 {
1329     OptixTraversableHandle child;
1330     unsigned int pad[2];
1331     float transform[12];

```

```

1404
1407     float invTransform[12];
1408 } OptixStaticTransform;
1409
1434 typedef struct OptixMatrixMotionTransform
1435 {
1437     OptixTraversableHandle child;
1438
1441     OptixMotionOptions motionOptions;
1442
1444     unsigned int pad[3];
1445
1447     float transform[2][12];
1448 } OptixMatrixMotionTransform;
1449
1457 //      [ sx  a  b  pvx ]
1458 //  S = [  0  sy  c  pvz ]
1459 //      [  0  0  sz  pvz ]
1468 //      [  1  0  0  tx ]
1469 //  T = [  0  1  0  ty ]
1470 //      [  0  0  1  tz ]
1480 typedef struct OptixSRTData
1481 {
1484     float sx, a, b, pvx, sy, c, pvz, sz, pvz, qx, qy, qz, qw, tx, ty, tz;
1486 } OptixSRTData;
1487
1488 // TODO Define a static assert for C/pre-C++-11
1489 #if defined(__cplusplus) && __cplusplus >= 201103L
1490 static_assert(sizeof(OptixSRTData) == 16 * 4, "OptixSRTData has wrong size");
1491 #endif
1492
1517 typedef struct OptixSRTMotionTransform
1518 {
1520     OptixTraversableHandle child;
1521
1524     OptixMotionOptions motionOptions;
1525
1527     unsigned int pad[3];
1528
1530     OptixSRTData srtData[2];
1531 } OptixSRTMotionTransform;
1532
1533 // TODO Define a static assert for C/pre-C++-11
1534 #if defined(__cplusplus) && __cplusplus >= 201103L
1535 static_assert(sizeof(OptixSRTMotionTransform) == 8 + 12 + 12 + 2 * 16 * 4, "OptixSRTMotionTransform has
wrong size");
1536 #endif
1537
1541 typedef enum OptixTraversableType
1542 {
1544     OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM = 0x21C1,
1546     OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM = 0x21C2,
1548     OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM = 0x21C3,
1549 } OptixTraversableType;
1550
1554 typedef enum OptixPixelFormat
1555 {
1556     OPTIX_PIXEL_FORMAT_HALF1 = 0x220a,
1557     OPTIX_PIXEL_FORMAT_HALF2 = 0x2207,
1558     OPTIX_PIXEL_FORMAT_HALF3 = 0x2201,
1559     OPTIX_PIXEL_FORMAT_HALF4 = 0x2202,
1560     OPTIX_PIXEL_FORMAT_FLOAT1 = 0x220b,
1561     OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208,
1562     OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203,
1563     OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204,
1564     OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205,
1565     OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206,

```

```

1566     OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209
1567 } OptixPixelFormat;
1568
1572 typedef struct OptixImage2D
1573 {
1574     CUdeviceptr data;
1575     unsigned int width;
1576     unsigned int height;
1577     unsigned int rowStrideInBytes;
1578     unsigned int pixelStrideInBytes;
1579     OptixPixelFormat format;
1580 } OptixImage2D;
1581
1594 typedef enum OptixDenoiserModelKind
1595 {
1596     OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322,
1597     OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323,
1598     OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324,
1599     OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325,
1600     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326,
1601     OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327,
1602     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328
1603 } OptixDenoiserModelKind;
1604
1622 typedef struct OptixDenoiserOptions
1623 {
1624     // if nonzero, albedo image must be given in OptixDenoiserGuideLayer
1625     unsigned int guideAlbedo;
1626     // if nonzero, normal image must be given in OptixDenoiserGuideLayer
1627     unsigned int guideNormal;
1628 } OptixDenoiserOptions;
1629
1634 typedef struct OptixDenoiserGuideLayer
1635 {
1636     // albedo/bsdf image
1637     OptixImage2D albedo;
1638     // normal vector image (2d or 3d pixel format)
1639     OptixImage2D normal;
1640     // 2d flow image, pixel flow from previous to current frame for each pixel
1641     OptixImage2D flow;
1642     // Internal images used in temporal AOV denoising modes,
1643     // pixel format OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER
1644     OptixImage2D previousOutputInternalGuideLayer;
1645     OptixImage2D outputInternalGuideLayer;
1646     // 1d image specifying how trustworthy the flow vector at x,y position in
1647     // OptixDenoiserGuideLayer::flow is. Range 0..1 (low->high trustworthiness).
1648     // Ignored if data pointer in the image is zero.
1649     OptixImage2D flowTrustworthiness;
1650 } OptixDenoiserGuideLayer;
1651
1659 typedef enum OptixDenoiserAOVType
1660 {
1661     OPTIX_DENOISER_AOV_TYPE_NONE = 0,
1662     OPTIX_DENOISER_AOV_TYPE_BEAUTY = 0x7000,

```

```

1665     OPTIX_DENOISER_AOV_TYPE_SPECULAR    = 0x7001,
1666     OPTIX_DENOISER_AOV_TYPE_REFLECTION = 0x7002,
1667     OPTIX_DENOISER_AOV_TYPE_REFRACTION = 0x7003,
1668     OPTIX_DENOISER_AOV_TYPE_DIFFUSE    = 0x7004
1669 } OptixDenoiserAOVType;
1670
1671 typedef struct OptixDenoiserLayer
1672 {
1673     // input image (beauty or AOV)
1674     OptixImage2D input;
1675
1676     // denoised output image from previous frame if temporal model kind selected
1677     OptixImage2D previousOutput;
1678
1679     // denoised output for given input
1680     OptixImage2D output;
1681
1682     // Type of AOV, used in temporal AOV modes as a hint to improve image quality.
1683     OptixDenoiserAOVType type;
1684 } OptixDenoiserLayer;
1685
1686 typedef enum OptixDenoiserAlphaMode
1687 {
1688     OPTIX_DENOISER_ALPHA_MODE_COPY = 0,
1689     OPTIX_DENOISER_ALPHA_MODE_ALPHA_AS_AOV = 1,
1690     OPTIX_DENOISER_ALPHA_MODE_FULL_DENOISE_PASS = 2
1691 } OptixDenoiserAlphaMode;
1692
1693 typedef struct OptixDenoiserParams
1694 {
1695     OptixDenoiserAlphaMode denoiseAlpha;
1696
1697     CUdeviceptr hdrIntensity;
1698
1699     float blendFactor;
1700
1701     CUdeviceptr hdrAverageColor;
1702
1703     unsigned int temporalModeUsePreviousLayers;
1704 } OptixDenoiserParams;
1705
1706 typedef struct OptixDenoiserSizes
1707 {
1708     size_t stateSizeInBytes;
1709
1710     size_t withOverlapScratchSizeInBytes;
1711
1712     size_t withoutOverlapScratchSizeInBytes;
1713
1714     unsigned int overlapWindowSizeInPixels;
1715
1716     size_t computeAverageColorSizeInBytes;
1717
1718     size_t computeIntensitySizeInBytes;
1719
1720     size_t internalGuideLayerPixelSizeInBytes;
1721 } OptixDenoiserSizes;
1722
1723 typedef enum OptixRayFlags
1724 {
1725     OPTIX_RAY_FLAG_NONE = 0u,
1726
1727     OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u « 0,
1728
1729     OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u « 1,

```

```

1787
1790     OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u « 2,
1791
1793     OPTIX_RAY_FLAG_DISABLE_CLOSEST_HIT = 1u « 3,
1794
1799     OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u « 4,
1800
1805     OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u « 5,
1806
1812     OPTIX_RAY_FLAG_CULL_DISABLED_ANY_HIT = 1u « 6,
1813
1819     OPTIX_RAY_FLAG_CULL_ENFORCED_ANY_HIT = 1u « 7,
1820
1822     OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u « 10,
1823 } OptixRayFlags;
1824
1830 typedef enum OptixTransformType
1831 {
1832     OPTIX_TRANSFORM_TYPE_NONE = 0,
1833     OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM = 1,
1834     OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM = 2,
1835     OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM = 3,
1836     OPTIX_TRANSFORM_TYPE_INSTANCE = 4,
1837 } OptixTransformType;
1838
1841 typedef enum OptixTraversableGraphFlags
1842 {
1845     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0,
1846
1850     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u « 0,
1851
1856     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u « 1,
1857 } OptixTraversableGraphFlags;
1858
1862 typedef enum OptixCompileOptimizationLevel
1863 {
1865     OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0,
1867     OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340,
1869     OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341,
1871     OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342,
1873     OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343,
1874 } OptixCompileOptimizationLevel;
1875
1879 typedef enum OptixCompileDebugLevel
1880 {
1882     OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT = 0,
1884     OPTIX_COMPILE_DEBUG_LEVEL_NONE = 0x2350,
1887     OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL = 0x2351,
1889     OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353,
1891     OPTIX_COMPILE_DEBUG_LEVEL_FULL = 0x2352,
1892 } OptixCompileDebugLevel;
1893
1897 typedef enum OptixModuleCompileState
1898 {
1900     OPTIX_MODULE_COMPILE_STATE_NOT_STARTED = 0x2360,
1901
1903     OPTIX_MODULE_COMPILE_STATE_STARTED = 0x2361,
1904
1906     OPTIX_MODULE_COMPILE_STATE_PENDING_FAILURE = 0x2362,
1907
1909     OPTIX_MODULE_COMPILE_STATE_FAILED = 0x2363,
1910
1912     OPTIX_MODULE_COMPILE_STATE_COMPLETED = 0x2364,
1913 } OptixModuleCompileState;
1914
1915
1916

```

```

1949 typedef struct OptixModuleCompileBoundValueEntry {
1950     size_t pipelineParamOffsetInBytes;
1951     size_t sizeInBytes;
1952     const void* boundValuePtr;
1953     const char* annotation; // optional string to display, set to 0 if unused. If unused,
1954                             // OptiX will report the annotation as "No annotation"
1955 } OptixModuleCompileBoundValueEntry;
1956
1957 typedef enum OptixPayloadTypeID {
1958     OPTIX_PAYLOAD_TYPE_DEFAULT = 0,
1959     OPTIX_PAYLOAD_TYPE_ID_0 = (1 « 0u),
1960     OPTIX_PAYLOAD_TYPE_ID_1 = (1 « 1u),
1961     OPTIX_PAYLOAD_TYPE_ID_2 = (1 « 2u),
1962     OPTIX_PAYLOAD_TYPE_ID_3 = (1 « 3u),
1963     OPTIX_PAYLOAD_TYPE_ID_4 = (1 « 4u),
1964     OPTIX_PAYLOAD_TYPE_ID_5 = (1 « 5u),
1965     OPTIX_PAYLOAD_TYPE_ID_6 = (1 « 6u),
1966     OPTIX_PAYLOAD_TYPE_ID_7 = (1 « 7u)
1967 } OptixPayloadTypeID;
1968
1969 typedef enum OptixPayloadSemantics
1970 {
1971     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE = 0,
1972     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ = 1u « 0,
1973     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE = 2u « 0,
1974     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u « 0,
1975
1976     OPTIX_PAYLOAD_SEMANTICS_CH_NONE = 0,
1977     OPTIX_PAYLOAD_SEMANTICS_CH_READ = 1u « 2,
1978     OPTIX_PAYLOAD_SEMANTICS_CH_WRITE = 2u « 2,
1979     OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE = 3u « 2,
1980
1981     OPTIX_PAYLOAD_SEMANTICS_MS_NONE = 0,
1982     OPTIX_PAYLOAD_SEMANTICS_MS_READ = 1u « 4,
1983     OPTIX_PAYLOAD_SEMANTICS_MS_WRITE = 2u « 4,
1984     OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE = 3u « 4,
1985
1986     OPTIX_PAYLOAD_SEMANTICS_AH_NONE = 0,
1987     OPTIX_PAYLOAD_SEMANTICS_AH_READ = 1u « 6,
1988     OPTIX_PAYLOAD_SEMANTICS_AH_WRITE = 2u « 6,
1989     OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE = 3u « 6,
1990
1991     OPTIX_PAYLOAD_SEMANTICS_IS_NONE = 0,
1992     OPTIX_PAYLOAD_SEMANTICS_IS_READ = 1u « 8,
1993     OPTIX_PAYLOAD_SEMANTICS_IS_WRITE = 2u « 8,
1994     OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE = 3u « 8,
1995 } OptixPayloadSemantics;
1996
1997 typedef struct OptixPayloadType
1998 {
1999     unsigned int numPayloadValues;
2000
2001     const unsigned int *payloadSemantics;
2002 } OptixPayloadType;
2003
2004 typedef struct OptixModuleCompileOptions
2005 {
2006     int maxRegisterCount;
2007
2008     OptixCompileOptimizationLevel optLevel;
2009
2010     OptixCompileDebugLevel debugLevel;
2011
2012     const OptixModuleCompileBoundValueEntry* boundValues;
2013
2014     unsigned int numBoundValues;
2015 }

```

```

2044     unsigned int numPayloadTypes;
2045
2047     OptixPayloadType *payloadTypes;
2048
2049 } OptixModuleCompileOptions;
2050
2052 typedef enum OptixProgramGroupKind
2053 {
2054     OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421,
2055
2056     OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422,
2057
2058     OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423,
2059
2060     OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424,
2061
2062     OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425
2063 } OptixProgramGroupKind;
2064
2066 typedef enum OptixProgramGroupFlags
2067 {
2068     OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0
2069 } OptixProgramGroupFlags;
2070
2072 typedef struct OptixProgramGroupSingleModule
2073 {
2074     OptixModule module;
2075     const char* entryFunctionName;
2076 } OptixProgramGroupSingleModule;
2077
2079 typedef struct OptixProgramGroupHitgroup
2080 {
2081     OptixModule moduleCH;
2082     const char* entryFunctionNameCH;
2083     OptixModule moduleAH;
2084     const char* entryFunctionNameAH;
2085     OptixModule moduleIS;
2086     const char* entryFunctionNameIS;
2087 } OptixProgramGroupHitgroup;
2088
2090 typedef struct OptixProgramGroupCallables
2091 {
2092     OptixModule moduleDC;
2093     const char* entryFunctionNameDC;
2094     OptixModule moduleCC;
2095     const char* entryFunctionNameCC;
2096 } OptixProgramGroupCallables;
2097
2099 typedef struct OptixProgramGroupDesc
2100 {
2101     OptixProgramGroupKind kind;
2102
2103     unsigned int flags;
2104
2105     union
2106     {
2107         OptixProgramGroupSingleModule raygen;
2108         OptixProgramGroupSingleModule miss;
2109         OptixProgramGroupSingleModule exception;
2110         OptixProgramGroupCallables callables;
2111         OptixProgramGroupHitgroup hitgroup;
2112     };
2113 } OptixProgramGroupDesc;
2114
2116 typedef struct OptixProgramGroupOptions
2117 {
2118     OptixPayloadType* payloadType;

```

```

2176 } OptixProgramGroupOptions;
2177
2179 typedef enum OptixExceptionCodes
2180 {
2183     OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1,
2184
2187     OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2,
2188
2193     OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED = -3,
2194
2200     OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE = -5,
2201
2206     OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT = -6,
2207
2212     //      sbt-index (See optixGetExceptionInvalidSbtOffset),
2213     //      sbt-instance-offset (See OptixInstance::sbtOffset),
2224     OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT = -7,
2225
2228     OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE = -8,
2229
2234     OPTIX_EXCEPTION_CODE_INVALID_RAY = -9,
2235
2241     OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH = -10,
2242
2244     OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH = -11,
2245
2250     OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT = -12,
2251
2254     OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD = -13,
2255
2258     OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD = -14,
2259
2266     OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS = -15,
2267
2270     OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0 = -16,
2271     OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_1 = -17,
2272     OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2 = -18,
2273
2275     OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS = -32,
2276
2278     OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH = -33,
2279 } OptixExceptionCodes;
2280
2284 typedef enum OptixExceptionFlags
2285 {
2287     OPTIX_EXCEPTION_FLAG_NONE = 0,
2288
2290     OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u « 0,
2291
2293     OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u « 1,
2294
2297     OPTIX_EXCEPTION_FLAG_USER = 1u « 2,
2298
2300     OPTIX_EXCEPTION_FLAG_DEBUG = 1u « 3
2301 } OptixExceptionFlags;
2302
2308 typedef struct OptixPipelineCompileOptions
2309 {
2311     int usesMotionBlur;
2312
2314     unsigned int traversableGraphFlags;
2315
2318     int numPayloadValues;
2319
2322     int numAttributeValues;
2323
2325     unsigned int exceptionFlags;

```



```

2326
2330     const char* pipelineLaunchParamsVariableName;
2331
2334     unsigned int usesPrimitiveTypeFlags;
2335
2337     int allowOpacityMicromaps;
2338 } OptixPipelineCompileOptions;
2339
2343 typedef struct OptixPipelineLinkOptions
2344 {
2347     unsigned int maxTraceDepth;
2348
2349 } OptixPipelineLinkOptions;
2350
2354 typedef struct OptixShaderBindingTable
2355 {
2358     CUdeviceptr raygenRecord;
2359
2362     CUdeviceptr exceptionRecord;
2363
2367     CUdeviceptr missRecordBase;
2368     unsigned int missRecordStrideInBytes;
2369     unsigned int missRecordCount;
2371
2375     CUdeviceptr hitgroupRecordBase;
2376     unsigned int hitgroupRecordStrideInBytes;
2377     unsigned int hitgroupRecordCount;
2379
2384     CUdeviceptr callablesRecordBase;
2385     unsigned int callablesRecordStrideInBytes;
2386     unsigned int callablesRecordCount;
2388
2389 } OptixShaderBindingTable;
2390
2394 typedef struct OptixStackSizes
2395 {
2397     unsigned int cssRG;
2399     unsigned int cssMS;
2401     unsigned int cssCH;
2403     unsigned int cssAH;
2405     unsigned int cssIS;
2407     unsigned int cssCC;
2409     unsigned int dssDC;
2410
2411 } OptixStackSizes;
2412
2414 typedef enum OptixQueryFunctionTableOptions
2415 {
2417     OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0
2418
2419 } OptixQueryFunctionTableOptions;
2420
2422 typedef OptixResult(OptixQueryFunctionTable_t)(int          abiId,
2423  unsigned int numOptions,
2424  OptixQueryFunctionTableOptions* /*optionKeys*/,
2425  const void** /*optionValues*/,
2426  void* functionTable,
2427  size_t sizeOfTable);
2428
2433 typedef struct OptixBuiltinISOOptions
2434 {
2435     OptixPrimitiveType    builtinISModuleType;
2437     int                   usesMotionBlur;
2439     unsigned int          buildFlags;
2441     unsigned int          curveEndcapFlags;
2442 } OptixBuiltinISOOptions;
2443

```

```
2444 #if defined(__CUDACC__)
2449 typedef struct OptixInvalidRayExceptionDetails
2450 {
2451     float3 origin;
2452     float3 direction;
2453     float tmin;
2454     float tmax;
2455     float time;
2456 } OptixInvalidRayExceptionDetails;
2457
2464 typedef struct OptixParameterMismatchExceptionDetails
2465 {
2467     unsigned int expectedParameterCount;
2469     unsigned int passedArgumentCount;
2471     unsigned int sbtIndex;
2473     char* callableName;
2474 } OptixParameterMismatchExceptionDetails;
2475 #endif
2476
2477 // end group optix_types
2479
2480 #endif // OPTIX_OPTIX_TYPES_H
```

## 8.25 main.dox File Reference