



NVIDIA OptiX 8.1

API Reference Manual

22 October 2024
Version 8.1

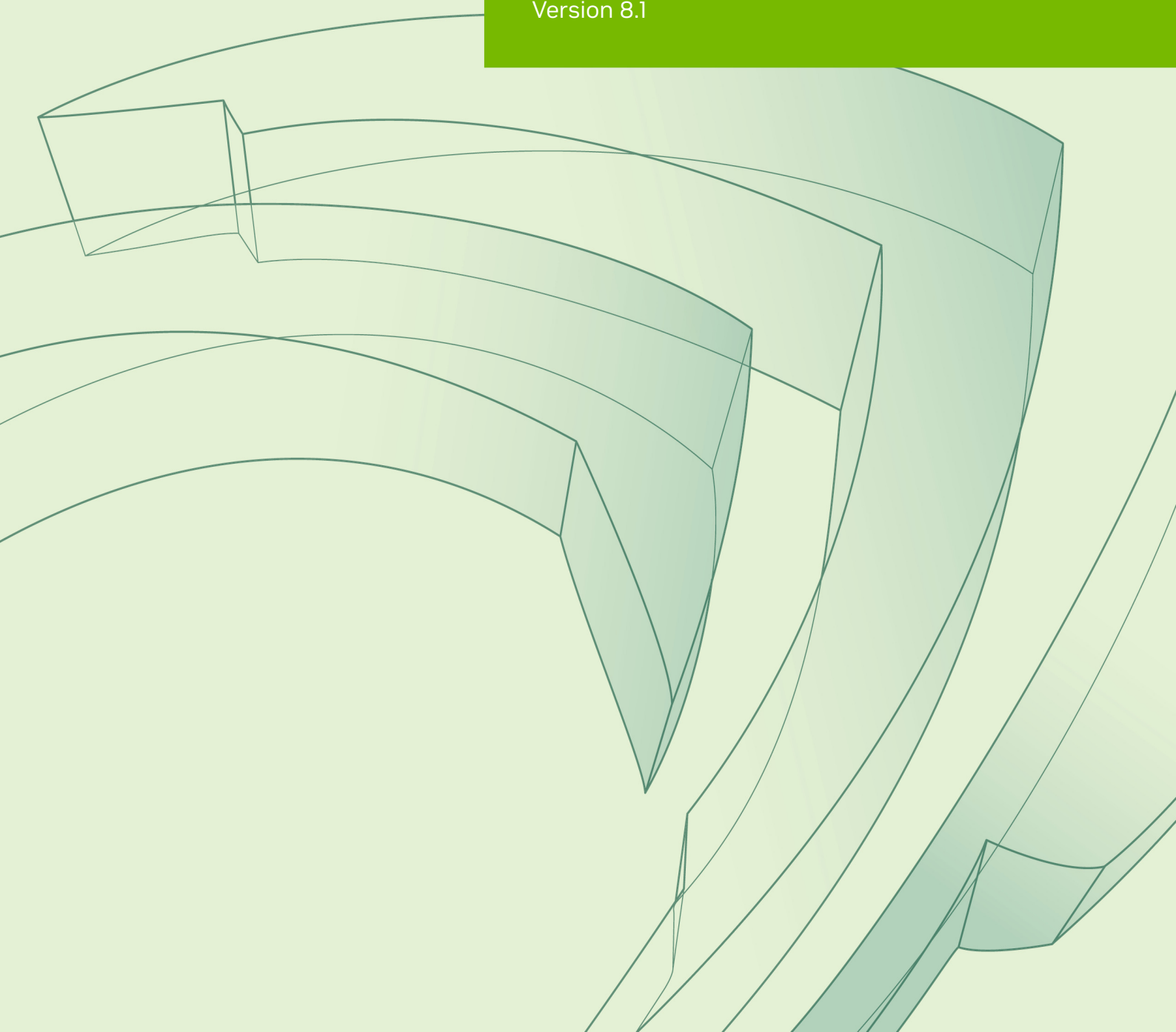


Table of Contents

1	NVIDIA OptiX 8.1 API	1
2	Module Index	1
2.1	Modules	1
3	Class Index	1
3.1	Class List	1
4	File Index	4
4.1	File List	4
5	Module Documentation	5
5.1	Device API	5
5.2	Function Table	54
5.3	Host API	55
5.4	Error handling	56
5.5	Device context	56
5.6	Pipelines	56
5.7	Modules	56
5.8	Tasks	56
5.9	Program groups	56
5.10	Launches	56
5.11	Acceleration structures	56
5.12	Denoiser	56
5.13	Utilities	56
5.14	Types	63
6	Namespace Documentation	112
6.1	optix_impl Namespace Reference	112
6.2	optix_internal Namespace Reference	116
7	Class Documentation	116
7.1	OptixAabb Struct Reference	116
7.2	OptixAccelBufferSizes Struct Reference	117
7.3	OptixAccelBuildOptions Struct Reference	118
7.4	OptixAccelEmitDesc Struct Reference	119
7.5	OptixBuildInput Struct Reference	119
7.6	OptixBuildInputCurveArray Struct Reference	120
7.7	OptixBuildInputCustomPrimitiveArray Struct Reference	123
7.8	OptixBuildInputDisplacementMicromap Struct Reference	124
7.9	OptixBuildInputInstanceArray Struct Reference	126
7.10	OptixBuildInputOpacityMicromap Struct Reference	127
7.11	OptixBuildInputSphereArray Struct Reference	129
7.12	OptixBuildInputTriangleArray Struct Reference	131
7.13	OptixBuiltinISOOptions Struct Reference	133
7.14	OptixDenoiserGuideLayer Struct Reference	134
7.15	OptixDenoiserLayer Struct Reference	135
7.16	OptixDenoiserOptions Struct Reference	135
7.17	OptixDenoiserParams Struct Reference	136
7.18	OptixDenoiserSizes Struct Reference	137
7.19	OptixDeviceContextOptions Struct Reference	138
7.20	OptixDisplacementMicromapArrayBuildInput Struct Reference	139
7.21	OptixDisplacementMicromapDesc Struct Reference	140

7.22	OptixDisplacementMicromapHistogramEntry Struct Reference	140
7.23	OptixDisplacementMicromapUsageCount Struct Reference	141
7.24	OptixFunctionTable Struct Reference	142
7.25	OptixImage2D Struct Reference	152
7.26	OptixInstance Struct Reference	153
7.27	OptixMatrixMotionTransform Struct Reference	154
7.28	OptixMicromapBuffers Struct Reference	155
7.29	OptixMicromapBufferSizes Struct Reference	155
7.30	OptixModuleCompileBoundValueEntry Struct Reference	156
7.31	OptixModuleCompileOptions Struct Reference	157
7.32	OptixMotionOptions Struct Reference	158
7.33	OptixOpacityMicromapArrayBuildInput Struct Reference	159
7.34	OptixOpacityMicromapDesc Struct Reference	160
7.35	OptixOpacityMicromapHistogramEntry Struct Reference	160
7.36	OptixOpacityMicromapUsageCount Struct Reference	161
7.37	OptixPayloadType Struct Reference	162
7.38	OptixPipelineCompileOptions Struct Reference	162
7.39	OptixPipelineLinkOptions Struct Reference	163
7.40	OptixProgramGroupCallables Struct Reference	164
7.41	OptixProgramGroupDesc Struct Reference	165
7.42	OptixProgramGroupHitgroup Struct Reference	166
7.43	OptixProgramGroupOptions Struct Reference	167
7.44	OptixProgramGroupSingleModule Struct Reference	167
7.45	OptixRelocateInput Struct Reference	168
7.46	OptixRelocateInputInstanceArray Struct Reference	169
7.47	OptixRelocateInputOpacityMicromap Struct Reference	169
7.48	OptixRelocateInputTriangleArray Struct Reference	169
7.49	OptixRelocationInfo Struct Reference	170
7.50	OptixShaderBindingTable Struct Reference	170
7.51	OptixSRTData Struct Reference	172
7.52	OptixSRTMotionTransform Struct Reference	174
7.53	OptixStackSizes Struct Reference	175
7.54	OptixStaticTransform Struct Reference	176
7.55	OptixUtilDenoiserImageTile Struct Reference	177
7.56	optix_internal::TypePack<... > Struct Template Reference	178
8	File Documentation	178
8.1	optix_device_impl.h File Reference	178
8.2	optix_device_impl.h	210
8.3	optix_device_impl_transformations.h File Reference	245
8.4	optix_device_impl_transformations.h	246
8.5	optix_micromap_impl.h File Reference	253
8.6	optix_micromap_impl.h	253
8.7	optix.h File Reference	256
8.8	optix.h	257
8.9	optix_denoiser_tiling.h File Reference	257
8.10	optix_denoiser_tiling.h	257
8.11	optix_device.h File Reference	262
8.12	optix_device.h	269
8.13	optix_function_table.h File Reference	278
8.14	optix_function_table.h	279
8.15	optix_function_table_definition.h File Reference	284
8.16	optix_function_table_definition.h	284

8.17	optix_host.h File Reference	285
8.18	optix_host.h	311
8.19	optix_micromap.h File Reference	316
8.20	optix_micromap.h	317
8.21	optix_stack_size.h File Reference	318
8.22	optix_stack_size.h	319
8.23	optix_stubs.h File Reference	323
8.24	optix_stubs.h	324
8.25	optix_types.h File Reference	336
8.26	optix_types.h	346
8.27	main.dox File Reference	365

1 NVIDIA OptiX 8.1 API

This document describes the NVIDIA OptiX 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	54
Host API	55
Error handling	56
Device context	56
Pipelines	56
Modules	56
Tasks	56
Program groups	56
Launches	56
Acceleration structures	56
Denoiser	56
Utilities	56
Types	63

3 Class Index

3.1 Class List

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

<code>OptixAabb</code>	
AABB inputs	116
<code>OptixAccelBufferSizes</code>	
Struct for querying builder allocation requirements	117
<code>OptixAccelBuildOptions</code>	
Build options for acceleration structures	118
<code>OptixAccelEmitDesc</code>	
Specifies a type and output destination for emitted post-build properties	119
<code>OptixBuildInput</code>	
Build inputs	119
<code>OptixBuildInputCurveArray</code>	
Curve inputs	120
<code>OptixBuildInputCustomPrimitiveArray</code>	
Custom primitive inputs	123

OptixBuildInputDisplacementMicromap	
Optional displacement part of a triangle array input	124
OptixBuildInputInstanceArray	
Instance and instance pointer inputs	126
OptixBuildInputOpacityMicromap	127
OptixBuildInputSphereArray	
Sphere inputs	129
OptixBuildInputTriangleArray	
Triangle inputs	131
OptixBuiltinISOptions	
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>	133
OptixDenoiserGuideLayer	
Guide layer for the denoiser	134
OptixDenoiserLayer	
Input/Output layers for the denoiser	135
OptixDenoiserOptions	
Options used by the denoiser	135
OptixDenoiserParams	
Various parameters used by the denoiser	136
OptixDenoiserSizes	
Various sizes related to the denoiser	137
OptixDeviceContextOptions	
Parameters used for <code>optixDeviceContextCreate()</code>	138
OptixDisplacementMicromapArrayBuildInput	
Inputs to displacement micromaps array construction	139
OptixDisplacementMicromapDesc	140
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 <code>OptixDisplacementMicromapUsageCount</code> , the histogram entry specifies how many displacement micromaps of a specific type are combined into a displacement micromap array	140
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 <code>OptixDisplacementMicromapHistogramEntry</code> , the usage count specifies how many displacement micromaps of a specific type are referenced by triangles in the AS	141
OptixFunctionTable	
The function table containing all API functions	142
OptixImage2D	
Image descriptor used by the denoiser	152

<code>OptixInstance</code>		
Instances		153
<code>OptixMatrixMotionTransform</code>		
Represents a matrix motion transformation		154
<code>OptixMicromapBuffers</code>		
Buffer inputs for opacity/displacement micromap array builds		155
<code>OptixMicromapBufferSizes</code>		
Conservative memory requirements for building a opacity/displacement micromap array		155
<code>OptixModuleCompileBoundValueEntry</code>		
Struct for specifying specializations for pipelineParams as specified in <code>OptixPipelineCompileOptions</code> ::pipelineLaunchParamsVariableName		156
<code>OptixModuleCompileOptions</code>		
Compilation options for module		157
<code>OptixMotionOptions</code>		
Motion options		158
<code>OptixOpacityMicromapArrayBuildInput</code>		
Inputs to opacity micromap array construction		159
<code>OptixOpacityMicromapDesc</code>		
Opacity micromap descriptor		160
<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		160
<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		161
<code>OptixPayloadType</code>		
Specifies a single payload type		162
<code>OptixPipelineCompileOptions</code>		
Compilation options for all modules of a pipeline		162
<code>OptixPipelineLinkOptions</code>		
Link options for a pipeline		163
<code>OptixProgramGroupCallables</code>		
Program group representing callables		164
<code>OptixProgramGroupDesc</code>		
Descriptor for program groups		165
<code>OptixProgramGroupHitgroup</code>		
Program group representing the hitgroup		166

<code>OptixProgramGroupOptions</code>		
Program group options		167
<code>OptixProgramGroupSingleModule</code>		
Program group representing a single module		167
<code>OptixRelocateInput</code>		
Relocation inputs		168
<code>OptixRelocateInputInstanceArray</code>		
Instance and instance pointer inputs		169
<code>OptixRelocateInputOpacityMicromap</code>		169
<code>OptixRelocateInputTriangleArray</code>		
Triangle inputs		169
<code>OptixRelocationInfo</code>		
Used to store information related to relocation of optix data structures		170
<code>OptixShaderBindingTable</code>		
Describes the shader binding table (SBT)		170
<code>OptixSRTData</code>		
Represents an SRT transformation		172
<code>OptixSRTMotionTransform</code>		
Represents an SRT motion transformation		174
<code>OptixStackSizes</code>		
Describes the stack size requirements of a program group		175
<code>OptixStaticTransform</code>		
Static transform		176
<code>OptixUtilDenoiserImageTile</code>		
Tile definition		177
<code>optix_internal::TypePack<... ></code>		178

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		178
<code>optix_device_impl_transformations.h</code>		
OptiX public API		245
<code>optix_micromap_impl.h</code>		
OptiX micromap helper functions		253
<code>optix.h</code>		
OptiX public API header		256
<code>optix_denoiser_tiling.h</code>		
OptiX public API header		257

optix_device.h	
OptiX public API header	262
optix_function_table.h	
OptiX public API header	278
optix_function_table_definition.h	
OptiX public API header	284
optix_host.h	
OptiX public API header	285
optix_micromap.h	
OptiX micromap helper functions	316
optix_stack_size.h	
OptiX public API header	318
optix_stubs.h	
OptiX public API header	323
optix_types.h	
OptiX public API header	336

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 optixTraverse (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)`
- `template<typename... Payload>`
`static __forceinline__ __device__ void optixTraverse (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 optixReorder (unsigned int coherenceHint, unsigned int numCoherenceHintBitsFromLSB)`
- `static __forceinline__ __device__ void optixReorder ()`
- `template<typename... Payload>`
`static __forceinline__ __device__ void optixInvoke (Payload &... payload)`
- `template<typename... Payload>`

- static `__forceinline__ __device__ void optixInvoke` (OptixPayloadTypeID type, Payload &... payload)
- `template<typename... RegAttributes>`
static `__forceinline__ __device__ void optixMakeHitObject` (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int SBTOffset, unsigned int SBTStride, unsigned int instIdx, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)
 - `template<typename... RegAttributes>`
static `__forceinline__ __device__ void optixMakeHitObject` (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int SBTOffset, unsigned int SBTStride, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)
 - `template<typename... RegAttributes>`
static `__forceinline__ __device__ void optixMakeHitObjectWithRecord` (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int sbtRecordIndex, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)
 - static `__forceinline__ __device__ void optixMakeMissHitObject` (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime)
 - static `__forceinline__ __device__ void optixMakeNopHitObject` ()
 - static `__forceinline__ __device__ bool optixHitObjectIsHit` ()
 - static `__forceinline__ __device__ bool optixHitObjectIsMiss` ()
 - static `__forceinline__ __device__ bool optixHitObjectIsNop` ()
 - static `__forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex` ()
 - 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 optixHitObjectGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixGetObjectRayOrigin ()
- static __forceinline__ __device__ float3 optixGetObjectRayDirection ()
- static __forceinline__ __device__ float optixGetRayTmin ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmin ()
- static __forceinline__ __device__ float optixGetRayTmax ()

- static __forceinline__ __device__ float optixHitObjectGetRayTmax ()
- static __forceinline__ __device__ float optixGetRayTime ()
- static __forceinline__ __device__ float optixHitObjectGetRayTime ()
- 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__ unsigned int optixHitObjectGetTransformListSize ()

- `static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle (unsigned int index)`
- `static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle (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 __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle (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__ unsigned int optixHitObjectGetAttribute_0 ()`

- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7 ()`
- `static __forceinline__ __device__ void optixTerminateRay ()`
- `static __forceinline__ __device__ void optixIgnoreIntersection ()`
- `static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex ()`
- `static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex ()`
- `static __forceinline__ __device__ unsigned int optixGetInstanceId ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId ()`
- `static __forceinline__ __device__ unsigned int optixGetInstanceIndex ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex ()`
- `static __forceinline__ __device__ unsigned int optixGetHitKind ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind ()`
- `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__ CUdeviceptr optixHitObjectGetSbtDataPointer ()`
- `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__ 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 make secondary `optixTrace` calls.

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 validation mode an exception will be generated.

Parameters

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

Available in RG, CH, MS, CC

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)`.

Direct callable programs are allowed to call `optixTrace`, but any secondary trace calls invoked from subsequently called CH, MS and callable programs will result an an error.

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 validation mode an exception will be generated.

Parameters

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

Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.3 `optixGetAttribute_0()`

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

Returns the attribute at the given slot index. There are up to 8 attributes available. The number of attributes is configured with `OptixPipelineCompileOptions::numAttributeValues`.

Available in AH, CH

5.1.2.4 `optixGetAttribute_1()`

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

5.1.2.5 `optixGetAttribute_2()`

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

5.1.2.6 `optixGetAttribute_3()`

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

5.1.2.7 `optixGetAttribute_4()`

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

5.1.2.8 `optixGetAttribute_5()`

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

5.1.2.9 `optixGetAttribute_6()`

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

5.1.2.10 `optixGetAttribute_7()`

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

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.

Available in all OptiX program types

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.

Available in all OptiX program types

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.

Available in all OptiX program types

5.1.2.14 optixGetCurveParameter()

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

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

Available in AH, CH

5.1.2.15 optixGetExceptionCode()

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

Returns the exception code.

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.

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()` Available in EX

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()` Available in EX

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()` Available in EX

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()` Available in EX

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()` Available in EX

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\(\)](#) Available in EX

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\(\)](#) Available in EX

5.1.2.24 [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 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.

Available in EX

5.1.2.25 [optixGetGASMotionStepCount\(\)](#)

```
static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (
    OptixTraversableHandle gas ) [static]
```

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

Available in all OptiX program types

5.1.2.26 [optixGetGASMotionTimeBegin\(\)](#)

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

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

Available in all OptiX program types

5.1.2.27 [optixGetGASMotionTimeEnd\(\)](#)

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

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

Available in all OptiX program types

5.1.2.28 [optixGetGASPointerFromHandle\(\)](#)

```
static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle (
    OptixTraversableHandle handle ) [static]
```

Returns a pointer to the geometry acceleration structure from its traversable handle.

Returns 0 if the traversable is not a geometry acceleration structure.

Available in all OptiX program types

5.1.2.29 optixGetGASTraversableHandle()

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

Returns the traversable handle for the Geometry Acceleration Structure (GAS) containing the current hit.

Available in IS, AH, CH

5.1.2.30 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 in AH and CH

5.1.2.31 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`.

Available in all OptiX program types

5.1.2.32 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 0

Available in IS, AH, CH

5.1.2.33 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`.

Available in all OptiX program types

5.1.2.34 `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

Available in IS, AH, CH

5.1.2.35 `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`.

Available in all OptiX program types

5.1.2.36 `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`.

Available in all OptiX program types

5.1.2.37 `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`.

Available in all OptiX program types

5.1.2.38 `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.

Available in all OptiX program types

5.1.2.39 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.

Available in all OptiX program types

5.1.2.40 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.

Available in all OptiX program types

5.1.2.41 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`.

Available in all OptiX program types

5.1.2.42 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.

Available in all OptiX program types

5.1.2.43 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.

Available in all OptiX program types

5.1.2.44 optixGetObjectRayDirection()

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

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

Available in IS and AH

5.1.2.45 optixGetObjectRayOrigin()

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

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

Available in IS and AH

5.1.2.46 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.

Available in IS, AH, CH

5.1.2.47 optixGetPayload_0()

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

Returns the 32-bit payload at the given slot index. There are up to 32 attributes available. The number of attributes is configured with `OptixPipelineCompileOptions::numPayloadValues` or with `OptixPayloadType` parameters set in `OptixModuleCompileOptions`.

Available in IS, AH, CH, MS

5.1.2.48 optixGetPayload_1()

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

5.1.2.49 optixGetPayload_10()

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

5.1.2.50 optixGetPayload_11()

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

5.1.2.51 optixGetPayload_12()

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

5.1.2.52 optixGetPayload_13()

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

5.1.2.53 optixGetPayload_14()

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

5.1.2.54 optixGetPayload_15()

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

5.1.2.55 optixGetPayload_16()

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

5.1.2.56 optixGetPayload_17()

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

5.1.2.57 optixGetPayload_18()

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

5.1.2.58 optixGetPayload_19()

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

5.1.2.59 optixGetPayload_20()

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

5.1.2.60 optixGetPayload_21()

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

5.1.2.61 optixGetPayload_22()

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

5.1.2.62 optixGetPayload_23()

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

5.1.2.63 optixGetPayload_24()

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

5.1.2.64 optixGetPayload_25()

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

5.1.2.65 optixGetPayload_25()

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

5.1.2.66 optixGetPayload_26()

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

5.1.2.67 optixGetPayload_27()

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

5.1.2.68 optixGetPayload_28()

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

5.1.2.69 optixGetPayload_29()

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

5.1.2.70 optixGetPayload_3()

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

5.1.2.71 optixGetPayload_30()

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

5.1.2.72 optixGetPayload_31()

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

5.1.2.73 optixGetPayload_4()

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

5.1.2.74 optixGetPayload_5()

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

5.1.2.75 optixGetPayload_6()

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

5.1.2.76 optixGetPayload_7()

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

5.1.2.77 optixGetPayload_8()

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

5.1.2.78 optixGetPayload_9()

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

5.1.2.79 `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.

Available in IS, AH, CH

5.1.2.80 `optixGetPrimitiveType()` [1/2]

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

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

Available in AH, CH

5.1.2.81 `optixGetPrimitiveType()` [2/2]

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

Function interpreting the result of `optixGetHitKind()`.

Available in all OptiX program types

5.1.2.82 `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.

Available in all OptiX program types

5.1.2.83 optixGetRayFlags()

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

Returns the rayFlags passed into optixTrace.

Available in IS, AH, CH, MS

5.1.2.84 optixGetRayTime()

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

Returns the rayTime passed into optixTrace.

Returns 0 if motion is disabled.

Available in IS, AH, CH, MS

5.1.2.85 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

Available in IS, AH, CH, MS

5.1.2.86 optixGetRayTmin()

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

Returns the tmin passed into optixTrace.

Available in IS, AH, CH, MS

5.1.2.87 optixGetRayVisibilityMask()

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

Returns the visibilityMask passed into optixTrace.

Available in IS, AH, CH, MS

5.1.2.88 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.

Available in all OptiX program types

5.1.2.89 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`.

Available in AH, CH

5.1.2.90 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.

Available in all OptiX program types

5.1.2.91 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.

Note that `optixGetSbtDataPointer` is not available in OptiX-enabled functions, because there is no SBT entry associated with the function.

Available in RG, IS, AH, CH, MS, EX, DC, CC

5.1.2.92 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.

Available in IS, AH, CH

5.1.2.93 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.

Available in all OptiX program types

5.1.2.94 `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`.

Available in all OptiX program types

5.1.2.95 `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`.

Available in all OptiX program types

5.1.2.96 `optixGetTransformListHandle()`

```

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

```

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

Available in IS, AH, CH

5.1.2.97 `optixGetTransformListSize()`

```

static __forceinline__ __device__ unsigned int optixGetTransformListSize ( )
[static]

```

Returns the number of transforms on the current transform list.

Available in IS, AH, CH

5.1.2.98 optixGetTransformTypeFromHandle()

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

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

Available in all OptiX program types

5.1.2.99 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.

Available in AH, CH

5.1.2.100 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.

Available in all OptiX program types

5.1.2.101 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.

Available in IS, AH, CH, MS

5.1.2.102 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.

Available in IS, AH, CH, MS

5.1.2.103 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.

Available in IS, AH, CH

5.1.2.104 optixHitObjectGetAttribute_0()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0
( ) [static]
```

Return the attribute at the given slot index for the current outgoing hit object. There are up to 8 attributes available. The number of attributes is configured with [OptixPipelineCompileOptions::numAttributeValues](#).

Results are undefined if the hit object is a miss.

Available in RG, CH, MS, CC, DC

5.1.2.105 optixHitObjectGetAttribute_1()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1
( ) [static]
```

5.1.2.106 optixHitObjectGetAttribute_2()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2
( ) [static]
```

5.1.2.107 optixHitObjectGetAttribute_3()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3
( ) [static]
```

5.1.2.108 optixHitObjectGetAttribute_4()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4
( ) [static]
```

5.1.2.109 optixHitObjectGetAttribute_5()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5
( ) [static]
```

5.1.2.110 optixHitObjectGetAttribute_6()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6
( ) [static]
```

5.1.2.111 optixHitObjectGetAttribute_7()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7
( ) [static]
```

5.1.2.112 optixHitObjectGetHitKind()

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

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

Results are undefined if the hit object is a miss.

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

Available in RG, CH, MS, CC, DC

5.1.2.113 optixHitObjectGetInstanceId()

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

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

Results are undefined if the hit object is a miss.

See [optixGetInstanceId\(\)](#).

Available in RG, CH, MS, CC, DC

5.1.2.114 optixHitObjectGetInstanceIndex()

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

Returns the zero-based index of the instance within its instance acceleration structure associated with the outgoing hit object.

Results are undefined if the hit object is a miss.

See [optixGetInstanceIndex\(\)](#).

Available in RG, CH, MS, CC, DC

5.1.2.115 optixHitObjectGetPrimitiveIndex()

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

Return the primitive index associated with the current outgoing hit object.

Results are undefined if the hit object is a miss.

See [optixGetPrimitiveIndex\(\)](#) for more details.

Available in RG, CH, MS, CC, DC

5.1.2.116 optixHitObjectGetRayTime()

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

Returns the rayTime passed into [optixTraverse](#), [optixMakeHitObject](#), [optixMakeHitObjectWithRecord](#), or [optixMakeMissHitObject](#).

Returns 0 for nop hit objects or when motion is disabled.

Available in RG, CH, MS, CC, DC

5.1.2.117 `optixHitObjectGetRayTmax()`

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

If the hit object is a hit, returns the smallest reported hitT.

If the hit object is a miss, returns the tmax passed into `optixTraverse` or `optixMakeMissHitObject`.

Returns 0 for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.118 `optixHitObjectGetRayTmin()`

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

Returns the tmin passed into `optixTraverse`, `optixMakeHitObject`, `optixMakeHitObjectWithRecord`, or `optixMakeMissHitObject`.

Returns 0.0f for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.119 `optixHitObjectGetSbtDataPointer()`

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

Device pointer address for the SBT associated with the hit or miss program for the current outgoing hit object.

Returns 0 for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.120 `optixHitObjectGetSbtGASIndex()`

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

Return the SBT GAS index of the closest intersected primitive associated with the current outgoing hit object.

Results are undefined if the hit object is a miss.

See `optixGetSbtGASIndex()` for details on the version for the incoming hit object.

Available in RG, CH, MS, CC, DC

5.1.2.121 `optixHitObjectGetSbtRecordIndex()`

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

Returns the SBT record index associated with the hit or miss program for the current outgoing hit object.

Returns 0 for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.122 `optixHitObjectGetTransformListHandle()`

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

Returns the traversable handle for a transform in the current transform list associated with the outgoing hit object.

Results are undefined if the hit object is a miss.

See `optixGetTransformListHandle()`

Available in RG, CH, MS, CC, DC

5.1.2.123 `optixHitObjectGetTransformListSize()`

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

Returns the number of transforms associated with the current outgoing hit object's transform list.

Returns zero when there is no hit (miss and nop).

See `optixGetTransformListSize()`

Available in RG, CH, MS, CC, DC

5.1.2.124 `optixHitObjectGetWorldRayDirection()`

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

Returns the `rayDirection` passed into `optixTraverse`, `optixMakeHitObject`, `optixMakeHitObjectWithRecord`, or `optixMakeMissHitObject`.

Returns [0, 0, 0] for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.125 `optixHitObjectGetWorldRayOrigin()`

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

Returns the `rayOrigin` passed into `optixTraverse`, `optixMakeHitObject`, `optixMakeHitObjectWithRecord`, or `optixMakeMissHitObject`.

Returns [0, 0, 0] for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.126 `optixHitObjectIsHit()`

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

Returns true if the current outgoing hit object contains a hit.

Available in RG, CH, MS, CC, DC

5.1.2.127 `optixHitObjectIsMiss()`

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

Returns true if the current outgoing hit object contains a miss.

Available in RG, CH, MS, CC, DC

5.1.2.128 optixHitObjectIsNop()

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

Returns true if the current outgoing hit object contains neither a hit nor miss. If executed with `optixInvoke`, no operation will result. An implied nop hit object is always assumed to exist even if there are no calls such as `optixTraverse` to explicitly create one.

Available in RG, CH, MS, CC, DC

5.1.2.129 optixIgnoreIntersection()

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

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

Available in AH

5.1.2.130 optixInvoke() [1/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixInvoke (
    OptixPayloadTypeID type,
    Payload &... payload ) [static]
```

Invokes `closesthit`, `miss` or `nop` based on the current outgoing hit object. After execution the current outgoing hit object will be set to `nop`. An implied `nop` hit object is always assumed to exist even if there are no calls to `optixTraverse`, `optixMakeHitObject`, `optixMakeMissHitObject`, or `optixMakeNopHitObject`.

Parameters

in	<i>type</i>	
in, out	<i>payload</i>	up to 32 unsigned int values that hold the payload

Available in RG, CH, MS, CC

5.1.2.131 optixInvoke() [2/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixInvoke (
    Payload &... payload ) [static]
```

Invokes `closesthit`, `miss` or `nop` based on the current outgoing hit object. After execution the current outgoing hit object will be set to `nop`. An implied `nop` hit object is always assumed to exist even if there are no calls to `optixTraverse`, `optixMakeHitObject`, `optixMakeMissHitObject`, or `optixMakeNopHitObject`.

Parameters

in, out	<i>payload</i>	up to 32 unsigned int values that hold the payload
---------	----------------	--

Available in RG, CH, MS, CC

5.1.2.132 `optixIsBackFaceHit()` [1/2]

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

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

Available in AH, CH

5.1.2.133 `optixIsBackFaceHit()` [2/2]

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

Function interpreting the result of `optixGetHitKind()`.

Available in all OptiX program types

5.1.2.134 `optixIsDisplacedMicromeshTriangleBackFaceHit()`

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

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

Available in AH, CH

5.1.2.135 `optixIsDisplacedMicromeshTriangleFrontFaceHit()`

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

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

Available in AH, CH

5.1.2.136 `optixIsDisplacedMicromeshTriangleHit()`

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

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

Available in AH, CH

5.1.2.137 `optixIsFrontFaceHit()` [1/2]

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

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

Available in AH, CH

5.1.2.138 `optixIsFrontFaceHit()` [2/2]

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

Function interpreting the result of `optixGetHitKind()`.

Available in all OptiX program types

5.1.2.139 `optixIsTriangleBackFaceHit()`

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

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

Available in AH, CH

5.1.2.140 `optixIsTriangleFrontFaceHit()`

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

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

Available in AH, CH

5.1.2.141 `optixIsTriangleHit()`

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

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

Available in AH, CH

5.1.2.142 `optixMakeHitObject()` [1/2]

```
template<typename... RegAttributes>
static __forceinline__ __device__ void optixMakeHitObject (
    OptixTraversableHandle handle,
    float3 rayOrigin,
    float3 rayDirection,
    float tmin,
    float tmax,
    float rayTime,
    unsigned int SBTOffset,
    unsigned int SBTstride,
    unsigned int instIdx,
    const OptixTraversableHandle * transforms,
    unsigned int numTransforms,
    unsigned int sbtGASIdx,
    unsigned int primIdx,
    unsigned int hitKind,
    RegAttributes... regAttributes ) [static]
```

Constructs an outgoing hit object from the hit information provided. This hit object will now become the current outgoing hit object and will overwrite the current outgoing hit object. This method includes the ability to specify arbitrary numbers of `OptixTraversableHandle` pointers for scenes with 0 to `OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH` levels of transforms.

Parameters

in	<i>handle</i>	
in	<i>rayOrigin</i>	

Parameters

in	<i>rayDirection</i>	
in	<i>tmin</i>	
in	<i>tmax</i>	
in	<i>rayTime</i>	
in	<i>SBToffset</i>	really only 4 bits
in	<i>SBTstride</i>	really only 4 bits
in	<i>instIdx</i>	
in	<i>transforms</i>	
in	<i>numTransforms</i>	
in	<i>sbtGASIdx</i>	
in	<i>primIdx</i>	
in	<i>hitKind</i>	
in	<i>regAttributes</i>	up to 8 attribute registers

Available in RG, CH, MS, CC

5.1.2.143 optixMakeHitObject() [2/2]

```
template<typename... RegAttributes>
static __forceinline__ __device__ void optixMakeHitObject (
    OptixTraversableHandle handle,
    float3 rayOrigin,
    float3 rayDirection,
    float tmin,
    float tmax,
    float rayTime,
    unsigned int SBToffset,
    unsigned int SBTstride,
    unsigned int instIdx,
    unsigned int sbtGASIdx,
    unsigned int primIdx,
    unsigned int hitKind,
    RegAttributes... regAttributes ) [static]
```

Constructs an outgoing hit object from the hit information provided. This hit object will now become the current outgoing hit object and will overwrite the current outgoing hit object.

Parameters

in	<i>handle</i>	
in	<i>rayOrigin</i>	
in	<i>rayDirection</i>	
in	<i>tmin</i>	

Parameters

in	<i>tmax</i>	
in	<i>rayTime</i>	
in	<i>SBTOffset</i>	really only 4 bits
in	<i>SBTstride</i>	really only 4 bits
in	<i>instIdx</i>	
in	<i>sbtGASIdx</i>	
in	<i>primIdx</i>	
in	<i>hitKind</i>	
in	<i>regAttributes</i>	up to 8 attribute registers

Available in RG, CH, MS, CC

5.1.2.144 optixMakeHitObjectWithRecord()

```
template<typename... RegAttributes>
static __forceinline__ __device__ void optixMakeHitObjectWithRecord (
    OptixTraversableHandle handle,
    float3 rayOrigin,
    float3 rayDirection,
    float tmin,
    float tmax,
    float rayTime,
    unsigned int sbtRecordIndex,
    unsigned int instIdx,
    const OptixTraversableHandle * transforms,
    unsigned int numTransforms,
    unsigned int sbtGASIdx,
    unsigned int primIdx,
    unsigned int hitKind,
    RegAttributes... regAttributes ) [static]
```

Constructs an outgoing hit object from the hit information provided. The SBT record index is explicitly specified. This hit object will now become the current outgoing hit object and will overwrite the current outgoing hit object.

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>	

Parameters

in	<i>sbtRecordIndex</i>	32 bits
in	<i>instIdx</i>	
in	<i>transforms</i>	
in	<i>numTransforms</i>	
in	<i>sbtGASIdx</i>	
in	<i>primIdx</i>	
in	<i>hitKind</i>	
in	<i>regAttributes</i>	up to 8 attribute registers

Available in RG, CH, MS, CC

5.1.2.145 optixMakeMissHitObject()

```
static __forceinline__ __device__ void optixMakeMissHitObject (
    unsigned int missSBTIndex,
    float3 rayOrigin,
    float3 rayDirection,
    float tmin,
    float tmax,
    float rayTime ) [static]
```

Constructs an outgoing hit object from the miss information provided. The SBT record index is explicitly specified as an argument. This hit object will now become the current outgoing hit object and will overwrite the current outgoing hit object.

Parameters

in	<i>missSBTIndex</i>	specifies the miss program invoked on a miss
in	<i>rayOrigin</i>	
in	<i>rayDirection</i>	
in	<i>tmin</i>	
in	<i>tmax</i>	
in	<i>rayTime</i>	

Available in RG, CH, MS, CC

5.1.2.146 optixMakeNopHitObject()

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

Constructs an outgoing hit object that when invoked does nothing (neither the miss nor the closest hit shader will be invoked). This hit object will now become the current outgoing hit object and will overwrite the current outgoing hit object. Accessors such as `optixHitObjectGetInstanceId` will return 0 or 0 filled structs. Only `optixHitObjectGetIsNop()` will return a non-zero result.

Available in RG, CH, MS, CC

5.1.2.147 `optixReorder()` [1/2]

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

Reorder the current thread using the hit object only, ie without further coherence hints.

Available in RG

5.1.2.148 `optixReorder()` [2/2]

```
static __forceinline__ __device__ void optixReorder (
    unsigned int coherenceHint,
    unsigned int numCoherenceHintBitsFromLSB ) [static]
```

Reorder the current thread using the current outgoing hit object and the coherence hint bits provided. Note that the coherence hint will take away some of the bits used in the hit object for sorting, so care should be made to reduce the number of hint bits as much as possible. Nop hit objects can use more coherence hint bits. Bits are taken from the lowest significant bit range. The maximum value of `numCoherenceHintBitsFromLSB` is implementation defined and can vary.

Parameters

in	<i>coherenceHint</i>
in	<i>numCoherenceHintBitsFromLSB</i>

Available in RG

5.1.2.149 `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>

Available in IS

5.1.2.150 `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\(float,unsigned int\)](#) Available in IS

5.1.2.151 `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\(float,unsigned int\)](#) Available in IS

5.1.2.152 `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\(float,unsigned int\)](#) Available in IS

5.1.2.153 `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\(float,unsigned int\)](#) Available in IS

5.1.2.154 `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\(float,unsigned int\)](#) Available in IS

5.1.2.155 `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\(float,unsigned int\)](#) Available in IS

5.1.2.156 `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\(float,unsigned int\)](#) Available in IS

5.1.2.157 `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)` Available in IS

5.1.2.158 `optixSetPayload_0()`

```

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

```

Writes the 32-bit payload at the given slot index. There are up to 32 attributes available. The number of attributes is configured with `OptixPipelineCompileOptions::numPayloadValues` or with `OptixPayloadType` parameters set in `OptixModuleCompileOptions`.

Available in IS, AH, CH, MS

5.1.2.159 `optixSetPayload_1()`

```

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

```

5.1.2.160 `optixSetPayload_10()`

```

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

```

5.1.2.161 `optixSetPayload_11()`

```

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

```

5.1.2.162 `optixSetPayload_12()`

```

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

```

5.1.2.163 `optixSetPayload_13()`

```

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

```

5.1.2.164 `optixSetPayload_14()`

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

5.1.2.165 `optixSetPayload_15()`

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

5.1.2.166 `optixSetPayload_16()`

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

5.1.2.167 `optixSetPayload_17()`

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

5.1.2.168 `optixSetPayload_18()`

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

5.1.2.169 `optixSetPayload_19()`

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

5.1.2.170 `optixSetPayload_2()`

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

5.1.2.171 `optixSetPayload_20()`

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

5.1.2.172 `optixSetPayload_21()`

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

5.1.2.173 `optixSetPayload_22()`

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

5.1.2.174 `optixSetPayload_23()`

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

```
    unsigned int p ) [static]
```

5.1.2.175 optixSetPayload_24()

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

5.1.2.176 optixSetPayload_25()

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

5.1.2.177 optixSetPayload_26()

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

5.1.2.178 optixSetPayload_27()

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

5.1.2.179 optixSetPayload_28()

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

5.1.2.180 optixSetPayload_29()

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

5.1.2.181 optixSetPayload_3()

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

5.1.2.182 optixSetPayload_30()

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

5.1.2.183 optixSetPayload_31()

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

5.1.2.184 optixSetPayload_4()

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

5.1.2.185 `optixSetPayload_5()`

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

5.1.2.186 `optixSetPayload_6()`

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

5.1.2.187 `optixSetPayload_7()`

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

5.1.2.188 `optixSetPayload_8()`

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

5.1.2.189 `optixSetPayload_9()`

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

5.1.2.190 `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.

Available in IS, AH, CH, MS

5.1.2.191 `optixTerminateRay()`

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

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

Available in AH

5.1.2.192 `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.

Available anywhere

5.1.2.193 `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.
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\(unsigned long long,unsigned int,float,float,unsigned int*\)](#) Available anywhere

5.1.2.194 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)

Parameters

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\(unsigned long long,unsigned int,float,float,unsigned int*\)](#) Available anywhere

5.1.2.195 [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.
-----------	----------------------	----------------------------------

Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.196 [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\(int\)](#) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.197 [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\(int\)](#) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.198 [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\(int\)](#) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.199 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\(int\)](#) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.200 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\(int\)](#) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.201 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\(int\)](#) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.202 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\(int\)](#) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.203 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\(int\)](#) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.204 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

Available in RG, CH, MS, CC

5.1.2.205 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>	

Parameters

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

Available in RG, CH, MS, CC

5.1.2.206 `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.

Available in IS, AH, CH

5.1.2.207 `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.

Available in IS, AH, CH

5.1.2.208 `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.

Available in IS, AH, CH

5.1.2.209 `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.

Available in IS, AH, CH

5.1.2.210 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.

Available in IS, AH, CH

5.1.2.211 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.

Available in IS, AH, CH

5.1.2.212 optixTraverse() [1/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixTraverse (
    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]
```

Similar to `optixTrace`, but does not invoke `closesthit` or `miss`. Instead, it overwrites the current outgoing hit object with the results of traversing the ray. The outgoing hit object may be invoked at some later

point with `optixInvoke`. The outgoing hit object can also be queried through various functions such as `optixHitObjectIsHit` or `optixHitObjectGetAttribute_0`.

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 <code>OptixRayFlags</code>
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

Available in RG, CH, MS, CC, DC

5.1.2.213 `optixTraverse()` [2/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixTraverse (
    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]
```

Similar to `optixTrace`, but does not invoke `closesthit` or `miss`. Instead, it overwrites the current outgoing hit object with the results of traversing the ray. The outgoing hit object may be invoked at some later point with `optixInvoke`. The outgoing hit object can also be queried through various functions such as `optixHitObjectIsHit` or `optixHitObjectGetAttribute_0`.

Parameters

in	<i>handle</i>	
----	---------------	--

Parameters

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

Available in RG, CH, MS, CC, DC

5.1.2.214 optixUndefinedValue()

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

Returns an undefined value.

Available anywhere

5.2 Function Table

Classes

- struct [OptixFunctionTable](#)

Macros

- `#define OPTIX_CONCATENATE_ABI_VERSION(prefix, macro) OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro)`
- `#define OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro) prefix ## _ ## macro`
- `#define OPTIX_FUNCTION_TABLE_SYMBOL OPTIX_CONCATENATE_ABI_VERSION(g_optixFunctionTable, OPTIX_ABI_VERSION)`

Typedefs

- `typedef struct OptixFunctionTable OptixFunctionTable`

Variables

- `OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL`

5.2.1 Detailed Description

OptiX Function Table.

5.2.2 Macro Definition Documentation

5.2.2.1 OPTIX_CONCATENATE_ABI_VERSION

```
#define OPTIX_CONCATENATE_ABI_VERSION(  
    prefix,  
    macro ) OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro)
```

5.2.2.2 OPTIX_CONCATENATE_ABI_VERSION_IMPL

```
#define OPTIX_CONCATENATE_ABI_VERSION_IMPL(  
    prefix,  
    macro ) prefix ## _ ## macro
```

5.2.2.3 OPTIX_FUNCTION_TABLE_SYMBOL

```
#define OPTIX_FUNCTION_TABLE_SYMBOL OPTIX_CONCATENATE_ABI_VERSION(g_  
optixFunctionTable, OPTIX_ABI_VERSION)
```

5.2.3 Typedef Documentation

5.2.3.1 OptixFunctionTable

```
typedef struct OptixFunctionTable OptixFunctionTable
```

The function table containing all API functions.

See `optixInit()` and `optixInitWithHandle()`.

5.2.4 Variable Documentation

5.2.4.1 OPTIX_FUNCTION_TABLE_SYMBOL

```
OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL
```

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.

Mixing multiple SDKs in a single application will result in symbol collisions. To enable different compilation units to use different SDKs, use `OPTIX_ENABLE_SDK_MIXING`.

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`

Macros

- `#define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline`
- `#define OPTIX_MICROMAP_FLOAT2_SUB(a, b) { a.x - b.x, a.y - b.y }`

Functions

- `OPTIX_MICROMAP_INLINE_FUNC float optix_impl::_uint_as_float (unsigned int x)`
- `OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::extractEvenBits (unsigned int x)`
- `OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::prefixEor (unsigned int x)`
- `OPTIX_MICROMAP_INLINE_FUNC void optix_impl::index2dbary (unsigned int index, unsigned int &u, unsigned int &v, unsigned int &w)`
- `OPTIX_MICROMAP_INLINE_FUNC void optix_impl::micro2bary (unsigned int index, unsigned int subdivisionLevel, float2 &bary0, float2 &bary1, float2 &bary2)`
- `OPTIX_MICROMAP_INLINE_FUNC float2 optix_impl::base2micro (const float2 &baseBarycentrics, const float2 microVertexBaseBarycentrics[3])`
- `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)
- OPTIXAPI `OptixResult optixInitWithHandle` (void **handlePtr)
- OPTIXAPI `OptixResult optixInit` (void)
- OPTIXAPI `OptixResult optixUninitWithHandle` (void *handle)

5.13.1 Detailed Description

OptiX Utilities.

5.13.2 Macro Definition Documentation

5.13.2.1 OPTIX_MICROMAP_FLOAT2_SUB

```
#define OPTIX_MICROMAP_FLOAT2_SUB(
    a,
    b ) { a.x - b.x, a.y - b.y }
```

5.13.2.2 OPTIX_MICROMAP_INLINE_FUNC

```
#define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline
```

5.13.3 Function Documentation

5.13.3.1 __uint_as_float()

```
OPTIX_MICROMAP_INLINE_FUNC float optix_impl::__uint_as_float (
    unsigned int x )
```

5.13.3.2 base2micro()

```
OPTIX_MICROMAP_INLINE_FUNC float2 optix_impl::base2micro (
    const float2 & baseBarycentrics,
    const float2 microVertexBaseBarycentrics[3] )
```

5.13.3.3 extractEvenBits()

```
OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::extractEvenBits (
    unsigned int x )
```

5.13.3.4 index2dbary()

```
OPTIX_MICROMAP_INLINE_FUNC void optix_impl::index2dbary (
    unsigned int index,
    unsigned int & u,
    unsigned int & v,
    unsigned int & w )
```

5.13.3.5 micro2bary()

```
OPTIX_MICROMAP_INLINE_FUNC void optix_impl::micro2bary (
    unsigned int index,
    unsigned int subdivisionLevel,
    float2 & bary0,
    float2 & bary1,
    float2 & bary2 )
```

5.13.3.6 optixInit()

```
OPTIXAPI 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.

5.13.3.7 optixInitWithHandle()

```
OPTIXAPI 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](#)

5.13.3.8 optixUninitWithHandle()

```
OPTIXAPI 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](#)

5.13.3.9 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.

5.13.3.10 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.

5.13.3.11 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.
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.

5.13.3.12 `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.
----	-------------------	--

Parameters

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.
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.

5.13.3.13 `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 maxims 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.

5.13.3.14 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>

5.13.3.15 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
in	<i>output</i>	full resolution output image
in	<i>overlapWindowSizeInPixels</i>	see OptixDenoiserSizes , optixDenoiserComputeMemoryResources
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

5.13.3.16 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

5.13.3.17 prefixEor()

```

OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::prefixEor (
    unsigned int x )

```

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 OptixDevicePropertyShaderExecutionReorderingFlags OptixDevicePropertyShaderExecutionReorderingFlags
- 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 enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
- typedef struct OptixDenoiserOptions OptixDenoiserOptions
- typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
- typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
- typedef struct OptixDenoiserLayer OptixDenoiserLayer
- 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 ,

- ```

OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING = 0x200A }

```
- enum OptixDeviceContextValidationMode {

```

OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0 ,
OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF }

```
  - enum OptixDevicePropertyShaderExecutionReorderingFlags {

```

OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_NONE = 0 ,
OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD = 1
<< 0 }

```
  - 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_BYTE3 = 0x2101 ,
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 OptixDenoiserAlphaMode {
  - OPTIX\_DENOISER\_ALPHA\_MODE\_COPY = 0 ,
  - OPTIX\_DENOISER\_ALPHA\_MODE\_DENOISE = 1 }
- 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 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 }

```
  - 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 }

```
  - 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 64ull
```

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
```

Alpha denoising mode.

See also `optixDenoiserCreate()`

#### 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
```

Various parameters used by the denoiser.

See also `optixDenoiserInvoke()`

`optixDenoiserComputeIntensity()`

`optixDenoiserComputeAverageColor()`

#### 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 OptixDevicePropertyShaderExecutionReorderingFlags

```
typedef enum OptixDevicePropertyShaderExecutionReorderingFlags
OptixDevicePropertyShaderExecutionReorderingFlags
```

Flags used to interpret the result of `optixDeviceContextGetProperty()` and `OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING`.

See also `optixDeviceContextGetProperty()`

### 5.14.3.36 OptixDisplacementMicromapArrayBuildInput

```
typedef struct OptixDisplacementMicromapArrayBuildInput
OptixDisplacementMicromapArrayBuildInput
```

Inputs to displacement micromaps array construction.

### 5.14.3.37 OptixDisplacementMicromapArrayIndexingMode

```
typedef enum OptixDisplacementMicromapArrayIndexingMode
OptixDisplacementMicromapArrayIndexingMode
```

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

### 5.14.3.38 OptixDisplacementMicromapBiasAndScaleFormat

```
typedef enum OptixDisplacementMicromapBiasAndScaleFormat
OptixDisplacementMicromapBiasAndScaleFormat
```

### 5.14.3.39 OptixDisplacementMicromapDesc

```
typedef struct OptixDisplacementMicromapDesc OptixDisplacementMicromapDesc
```

### 5.14.3.40 OptixDisplacementMicromapDirectionFormat

```
typedef enum OptixDisplacementMicromapDirectionFormat
OptixDisplacementMicromapDirectionFormat
```

### 5.14.3.41 OptixDisplacementMicromapFlags

```
typedef enum OptixDisplacementMicromapFlags OptixDisplacementMicromapFlags
```

Flags defining behavior of DMMs in a DMM array.

### 5.14.3.42 OptixDisplacementMicromapFormat

```
typedef enum OptixDisplacementMicromapFormat OptixDisplacementMicromapFormat
```

DMM input data format.

### 5.14.3.43 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.

#### 5.14.3.44 OptixDisplacementMicromapTriangleFlags

```
typedef enum OptixDisplacementMicromapTriangleFlags
OptixDisplacementMicromapTriangleFlags
```

#### 5.14.3.45 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.

#### 5.14.3.46 OptixExceptionCodes

```
typedef enum OptixExceptionCodes OptixExceptionCodes
```

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

#### 5.14.3.47 OptixExceptionFlags

```
typedef enum OptixExceptionFlags OptixExceptionFlags
```

Exception flags.

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

#### 5.14.3.48 OptixGeometryFlags

```
typedef enum OptixGeometryFlags OptixGeometryFlags
```

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

#### 5.14.3.49 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()`

#### 5.14.3.50 OptixImage2D

```
typedef struct OptixImage2D OptixImage2D
```

Image descriptor used by the denoiser.

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

#### 5.14.3.51 OptixIndicesFormat

```
typedef enum OptixIndicesFormat OptixIndicesFormat
```

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

### 5.14.3.52 OptixInstance

```
typedef struct OptixInstance OptixInstance
```

Instances.

See also `OptixBuildInputInstanceArray::instances`

### 5.14.3.53 OptixInstanceFlags

```
typedef enum OptixInstanceFlags OptixInstanceFlags
```

Flags set on the `OptixInstance::flags`.

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

### 5.14.3.54 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. |
| 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`

### 5.14.3.55 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\(\)](#)

#### 5.14.3.56 OptixMicromapBuffers

```
typedef struct OptixMicromapBuffers OptixMicromapBuffers
```

Buffer inputs for opacity/displacement micromap array builds.

#### 5.14.3.57 OptixMicromapBufferSizes

```
typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
```

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

#### 5.14.3.58 OptixModule

```
typedef struct OptixModule_t* OptixModule
```

Opaque type representing a module.

#### 5.14.3.59 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](#)

#### 5.14.3.60 OptixModuleCompileOptions

```
typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
```

Compilation options for module.

See also `optixModuleCreate()`

#### 5.14.3.61 OptixModuleCompileState

```
typedef enum OptixModuleCompileState OptixModuleCompileState
```

Module compilation state.

See also `optixModuleGetCompilationState()`, `optixModuleCreateWithTasks()`

#### 5.14.3.62 OptixMotionFlags

```
typedef enum OptixMotionFlags OptixMotionFlags
```

Enum to specify motion flags.

See also `OptixMotionOptions::flags`.

#### 5.14.3.63 OptixMotionOptions

```
typedef struct OptixMotionOptions OptixMotionOptions
```

Motion options.

See also `OptixAccelBuildOptions::motionOptions`, `OptixMatrixMotionTransform::motionOptions`, `OptixSRTMotionTransform::motionOptions`

#### 5.14.3.64 OptixOpacityMicromapArrayBuildInput

```
typedef struct OptixOpacityMicromapArrayBuildInput
OptixOpacityMicromapArrayBuildInput
```

Inputs to opacity micromap array construction.

#### 5.14.3.65 OptixOpacityMicromapArrayIndexingMode

```
typedef enum OptixOpacityMicromapArrayIndexingMode
OptixOpacityMicromapArrayIndexingMode
```

indexing mode of triangles to opacity micromaps in an array, used in `OptixBuildInputOpacityMicromap`.

#### 5.14.3.66 OptixOpacityMicromapDesc

```
typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc
```

Opacity micromap descriptor.

#### 5.14.3.67 OptixOpacityMicromapFlags

```
typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags
```

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

### 5.14.3.68 OptixOpacityMicromapFormat

```
typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat
```

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

### 5.14.3.69 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.

### 5.14.3.70 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.

### 5.14.3.71 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.

### 5.14.3.72 OptixPayloadType

```
typedef struct OptixPayloadType OptixPayloadType
```

Specifies a single payload type.

### 5.14.3.73 OptixPayloadTypeID

```
typedef enum OptixPayloadTypeID OptixPayloadTypeID
```

Payload type identifiers.

### 5.14.3.74 OptixPipeline

```
typedef struct OptixPipeline_t* OptixPipeline
```

Opaque type representing a pipeline.

#### 5.14.3.75 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()`

#### 5.14.3.76 OptixPipelineLinkOptions

```
typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
```

Link options for a pipeline.

See also `optixPipelineCreate()`

#### 5.14.3.77 OptixPixelFormat

```
typedef enum OptixPixelFormat OptixPixelFormat
```

Pixel formats used by the denoiser.

See also `OptixImage2D::format`

#### 5.14.3.78 OptixPrimitiveType

```
typedef enum OptixPrimitiveType OptixPrimitiveType
```

Builtin primitive types.

#### 5.14.3.79 OptixPrimitiveTypeFlags

```
typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags
```

Builtin flags may be bitwise combined.

See also `OptixPipelineCompileOptions::usesPrimitiveTypeFlags`

#### 5.14.3.80 OptixProgramGroup

```
typedef struct OptixProgramGroup_t* OptixProgramGroup
```

Opaque type representing a program group.

#### 5.14.3.81 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`

#### 5.14.3.82 OptixProgramGroupDesc

```
typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
```

Descriptor for program groups.

### 5.14.3.83 OptixProgramGroupFlags

```
typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
```

Flags for program groups.

### 5.14.3.84 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`

### 5.14.3.85 OptixProgramGroupKind

```
typedef enum OptixProgramGroupKind OptixProgramGroupKind
```

Distinguishes different kinds of program groups.

### 5.14.3.86 OptixProgramGroupOptions

```
typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
```

Program group options.

See also `optixProgramGroupCreate()`

### 5.14.3.87 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`

### 5.14.3.88 OptixQueryFunctionTable\_t

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

Type of the function `optixQueryFunctionTable()`

### 5.14.3.89 OptixQueryFunctionTableOptions

```
typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
```

Options that can be passed to `optixQueryFunctionTable()`

### 5.14.3.90 OptixRayFlags

```
typedef enum OptixRayFlags OptixRayFlags
```

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

See also `optixTrace()`

#### 5.14.3.91 `OptixRelocateInput`

```
typedef struct OptixRelocateInput OptixRelocateInput
```

Relocation inputs.

See also `optixAccelRelocate()`

#### 5.14.3.92 `OptixRelocateInputInstanceArray`

```
typedef struct OptixRelocateInputInstanceArray
OptixRelocateInputInstanceArray
```

Instance and instance pointer inputs.

See also `OptixRelocateInput::instanceArray`

#### 5.14.3.93 `OptixRelocateInputOpacityMicromap`

```
typedef struct OptixRelocateInputOpacityMicromap
OptixRelocateInputOpacityMicromap
```

#### 5.14.3.94 `OptixRelocateInputTriangleArray`

```
typedef struct OptixRelocateInputTriangleArray
OptixRelocateInputTriangleArray
```

Triangle inputs.

See also `OptixRelocateInput::triangleArray`

#### 5.14.3.95 `OptixRelocationInfo`

```
typedef struct OptixRelocationInfo OptixRelocationInfo
```

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

See also `optixOpacityMicromapArrayGetRelocationInfo()`, `optixOpacityMicromapArrayRelocate()`, `optixAccelGetRelocationInfo()`, `optixAccelRelocate()`, `optixCheckRelocationCompatibility()`

#### 5.14.3.96 `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()`

#### 5.14.3.97 `OptixShaderBindingTable`

```
typedef struct OptixShaderBindingTable OptixShaderBindingTable
```

Describes the shader binding table (SBT)

See also `optixLaunch()`

### 5.14.3.98 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  $qw = \cos(\text{theta}/2)$  and other components  $[qx, qy, qz] = \sin(\text{theta}/2) * [ax, ay, az]$  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()`

### 5.14.3.99 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()`

### 5.14.3.100 OptixStackSizes

```
typedef struct OptixStackSizes OptixStackSizes
```

Describes the stack size requirements of a program group.

See also `optixProgramGroupGetStackSize()`

#### 5.14.3.101 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()`

#### 5.14.3.102 OptixTask

```
typedef struct OptixTask_t* OptixTask
```

Opaque type representing a work task.

#### 5.14.3.103 OptixTransformFormat

```
typedef enum OptixTransformFormat OptixTransformFormat
```

Format of transform used in `OptixBuildInputTriangleArray::transformFormat`.

#### 5.14.3.104 OptixTransformType

```
typedef enum OptixTransformType OptixTransformType
```

Transform.

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

#### 5.14.3.105 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.

#### 5.14.3.106 OptixTraversableHandle

```
typedef unsigned long long OptixTraversableHandle
```

Traversable handle.

#### 5.14.3.107 OptixTraversableType

```
typedef enum OptixTraversableType OptixTraversableType
```

Traversable Handles.

See also `optixConvertPointerToTraversableHandle()`

#### 5.14.3.108 OptixVertexFormat

```
typedef enum OptixVertexFormat OptixVertexFormat
```

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

### 5.14.3.109 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

|                                               |                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPTIX_BUILD_FLAG_NONE                         | No special flags set.                                                                                                                                                                                                                                                                                                                                                                                |
| OPTIX_BUILD_FLAG_ALLOW_UPDATE                 | Allow updating the build with new vertex positions with subsequent calls to <code>optixAccelBuild</code> .                                                                                                                                                                                                                                                                                           |
| OPTIX_BUILD_FLAG_ALLOW_COMPACTION             |                                                                                                                                                                                                                                                                                                                                                                                                      |
| OPTIX_BUILD_FLAG_PREFER_FAST_TRACE            | This flag is mutually exclusive with <code>OPTIX_BUILD_FLAG_PREFER_FAST_BUILD</code> .                                                                                                                                                                                                                                                                                                               |
| OPTIX_BUILD_FLAG_PREFER_FAST_BUILD            | This flag is mutually exclusive with <code>OPTIX_BUILD_FLAG_PREFER_FAST_TRACE</code> .                                                                                                                                                                                                                                                                                                               |
| OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS   | Allow random access to build input vertices See <code>optixGetTriangleVertexData</code><br><code>optixGetLinearCurveVertexData</code><br><code>optixGetQuadraticBSplineVertexData</code><br><code>optixGetCubicBSplineVertexData</code><br><code>optixGetCatmullRomVertexData</code><br><code>optixGetRibbonVertexData</code><br><code>optixGetRibbonNormal</code> <code>optixGetSphereData</code> . |
| OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS | Allow random access to instances See <code>optixGetInstanceTraversableFromIAS</code> .                                                                                                                                                                                                                                                                                                               |

## Enumerator

|                                                  |                                                                                                                                                                                                                                                                                                                    |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE   | 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.       |
| OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS | If enabled, any instances referencing this GAS are allowed to disable the opacity micromap test through the DISABLE_OPACITY_MICROMAPS 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

|                                           |                                     |
|-------------------------------------------|-------------------------------------|
| <code>OPTIX_BUILD_OPERATION_BUILD</code>  | Perform a full build operation.     |
| <code>OPTIX_BUILD_OPERATION_UPDATE</code> | Perform an update using new bounds. |

#### 5.14.4.5 OptixCompileDebugLevel

enum `OptixCompileDebugLevel`

Debug levels.

See also `OptixModuleCompileOptions::debugLevel`

Enumerator

|                                                 |                                                                                                                             |
|-------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <code>OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT</code>  | Default currently is minimal.                                                                                               |
| <code>OPTIX_COMPILE_DEBUG_LEVEL_NONE</code>     | No debug information.                                                                                                       |
| <code>OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL</code>  | Generate information that does not impact performance. Note this replaces <code>OPTIX_COMPILE_DEBUG_LEVEL_LINEINFO</code> . |
| <code>OPTIX_COMPILE_DEBUG_LEVEL_MODERATE</code> | Generate some debug information with slight performance cost.                                                               |
| <code>OPTIX_COMPILE_DEBUG_LEVEL_FULL</code>     | Generate full debug information.                                                                                            |

#### 5.14.4.6 OptixCompileOptimizationLevel

enum `OptixCompileOptimizationLevel`

Optimization levels.

See also `OptixModuleCompileOptions::optLevel`

Enumerator

|                                                 |                                      |
|-------------------------------------------------|--------------------------------------|
| <code>OPTIX_COMPILE_OPTIMIZATION_DEFAULT</code> | Default is to run all optimizations. |
| <code>OPTIX_COMPILE_OPTIMIZATION_LEVEL_0</code> | No optimizations.                    |
| <code>OPTIX_COMPILE_OPTIMIZATION_LEVEL_1</code> | Some optimizations.                  |
| <code>OPTIX_COMPILE_OPTIMIZATION_LEVEL_2</code> | Most optimizations.                  |
| <code>OPTIX_COMPILE_OPTIMIZATION_LEVEL_3</code> | All optimizations.                   |

#### 5.14.4.7 OptixCurveEndcapFlags

enum `OptixCurveEndcapFlags`

Curve end cap types, for non-linear curves.

Enumerator

|                                         |                                                                               |
|-----------------------------------------|-------------------------------------------------------------------------------|
| <code>OPTIX_CURVE_ENDCAP_DEFAULT</code> | Default end caps. Round end caps for linear, no end caps for quadratic/cubic. |
|-----------------------------------------|-------------------------------------------------------------------------------|

Enumerator

|                       |                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------|
| 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`

Alpha denoising mode.

See also `optixDenoiserCreate()`

Enumerator

|                                   |                                                         |
|-----------------------------------|---------------------------------------------------------|
| OPTIX_DENOISER_ALPHA_MODE_COPY    | Copy alpha (if present) from input layer, no denoising. |
| OPTIX_DENOISER_ALPHA_MODE_DENOISE | Denoise alpha.                                          |

#### 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. |

## Enumerator

|                                              |                                                                                                                        |
|----------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| 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 <code>OptixPipelineLinkOptions::maxTraceDepth</code> . <code>sizeof(unsigned int)</code>                                                |
| OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH | Maximum value to pass into <code>optixPipelineSetStackSize</code> for parameter <code>maxTraversableGraphDepth</code> . <code>sizeof(unsigned int)</code> |
| 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). <code>sizeof(unsigned int)</code>    |

## Enumerator

|                                                               |                                                                                                                                                                                                                                          |
|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 <code>OptixInstance::instanceId</code> . sizeof(unsigned int)                                                                                                                                                      |
| OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK | The number of bits available for the <code>OptixInstance::visibilityMask</code> . 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)                                                                                                                |
| 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. sizeof(unsigned int)                                         |
| OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING             | Returns a flag specifying capabilities of the <code>optixReorder()</code> device function. See <code>OptixDevicePropertyShaderExecutionReorderingFlags</code> for documentation on the values that can be returned. sizeof(unsigned int) |

5.14.4.13 `OptixDevicePropertyShaderExecutionReorderingFlags`

enum `OptixDevicePropertyShaderExecutionReorderingFlags`

Flags used to interpret the result of `optixDeviceContextGetProperty()` and `OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING`.

See also `optixDeviceContextGetProperty()`

## Enumerator

|                                                                 |                                                                                                                                                                            |
|-----------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_NONE     | <code>optixReorder()</code> acts as a no-op, and no thread reordering is performed. Note that it is still legal to call this device function; no errors will be generated. |
| OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD |                                                                                                                                                                            |

5.14.4.14 `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. |

5.14.4.15 `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  |

5.14.4.16 `OptixDisplacementMicromapDirectionFormat`

enum `OptixDisplacementMicromapDirectionFormat`

## Enumerator

|                                                     |
|-----------------------------------------------------|
| OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_NONE   |
| OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_FLOAT3 |
| OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_HALF3  |

5.14.4.17 `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 <code>OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_BUILD</code> . |

Enumerator

|                                                    |                                                                                          |
|----------------------------------------------------|------------------------------------------------------------------------------------------|
| OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_BUILD | This flag is mutually exclusive with OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_TRACE. |
|----------------------------------------------------|------------------------------------------------------------------------------------------|

#### 5.14.4.18 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 |

#### 5.14.4.19 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.                                                                                                                                                                        |

#### 5.14.4.20 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.              |

### 5.14.4.21 OptixExceptionFlags

enum `OptixExceptionFlags`

Exception flags.

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

Enumerator

|                                                  |                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OPTIX_EXCEPTION_FLAG_NONE</code>           | No exception are enabled.                                                                                                                                                                                                                                                                                                                                                         |
| <code>OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW</code> | Enables exceptions check related to the continuation stack. This flag should be used when the application handles stack overflows in a user exception program as part of the normal flow of execution. For catching overflows during debugging and development, the device context validation mode should be used instead. See also <code>OptixDeviceContextValidationMode</code> |
| <code>OPTIX_EXCEPTION_FLAG_TRACE_DEPTH</code>    | Enables exceptions check related to trace depth. This flag should be used when the application handles trace depth overflows in a user exception program as part of the normal flow of execution. For catching overflows during debugging and development, the device context validation mode should be used instead. See also <code>OptixDeviceContextValidationMode</code>      |
| <code>OPTIX_EXCEPTION_FLAG_USER</code>           | Enables user exceptions via <code>optixThrowException()</code> . This flag must be specified for all modules in a pipeline if any module calls <code>optixThrowException()</code> .                                                                                                                                                                                               |

### 5.14.4.22 OptixGeometryFlags

enum `OptixGeometryFlags`

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

Enumerator

|                                                             |                                                                                                                                                                         |
|-------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OPTIX_GEOMETRY_FLAG_NONE</code>                       | No flags set.                                                                                                                                                           |
| <code>OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT</code>             | Disables the invocation of the anyhit program. Can be overridden by <code>OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT</code> and <code>OPTIX_RAY_FLAG_ENFORCE_ANYHIT</code> .    |
| <code>OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL</code> | 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. |

Enumerator

|                                                   |                                                                                                                                                                                         |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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. |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### 5.14.4.23 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.  |

#### 5.14.4.24 OptixIndicesFormat

enum `OptixIndicesFormat`

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

Enumerator

|                                      |                                                                                                                                 |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| OPTIX_INDICES_FORMAT_NONE            | No indices, this format must only be used in combination with triangle soups, i.e., <code>numIndexTriplets</code> must be zero. |
| OPTIX_INDICES_FORMAT_UNSIGNED_BYTE3  | Three bytes.                                                                                                                    |
| OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 | Three shorts.                                                                                                                   |
| OPTIX_INDICES_FORMAT_UNSIGNED_INT3   | Three ints.                                                                                                                     |

#### 5.14.4.25 OptixInstanceFlags

enum `OptixInstanceFlags`

Flags set on the `OptixInstance::flags`.

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

Enumerator

|                          |                      |
|--------------------------|----------------------|
| OPTIX_INSTANCE_FLAG_NONE | No special flag set. |
|--------------------------|----------------------|

## Enumerator

|                                                    |                                                                                                                                                                                                                                        |
|----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPTIX_INSTANCE_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.                                                |
| OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING           | Flip triangle orientation. This affects front/backface culling as well as the reported face in case of a hit.                                                                                                                          |
| OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT                 | Disable anyhit programs for all geometries of the instance. Can be overridden by OPTIX_RAY_FLAG_ENFORCE_ANYHIT. This flag is mutually exclusive with OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT.                                               |
| OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT                 | Enables anyhit programs for all geometries of the instance. Overrides OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT. Can be overridden by OPTIX_RAY_FLAG_DISABLE_ANYHIT. This flag is mutually exclusive with OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT. |
| OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE | Force 4-state opacity micromaps to behave as 2-state opacity micromaps during traversal.                                                                                                                                               |
| OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS      | Don't perform opacity micromap query for this instance. GAS must be built with ALLOW_DISABLE_OPACITY_MICROMAPS for this to be valid. This flag overrides FORCE_OPACITY_MICROMAP_2_STATE instance and ray flags.                        |

## 5.14.4.26 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_IMPENDING_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.   |

#### 5.14.4.27 OptixMotionFlags

enum `OptixMotionFlags`

Enum to specify motion flags.

See also `OptixMotionOptions::flags`.

Enumerator

|                                             |
|---------------------------------------------|
| <code>OPTIX_MOTION_FLAG_NONE</code>         |
| <code>OPTIX_MOTION_FLAG_START_VANISH</code> |
| <code>OPTIX_MOTION_FLAG_END_VANISH</code>   |

#### 5.14.4.28 OptixOpacityMicromapArrayIndexingMode

enum `OptixOpacityMicromapArrayIndexingMode`

indexing mode of triangles to opacity micromaps in an array, used in `OptixBuildInputOpacityMicromap`.

Enumerator

|                                                                 |                                                                                                                                                                                                                                                                                     |
|-----------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE</code>    | No opacity micromap is used.                                                                                                                                                                                                                                                        |
| <code>OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR</code>  | An implicit linear mapping of triangles to opacity micromaps in the opacity micromap array is used. <code>triangle[i]</code> will use <code>opacityMicromapArray[i]</code> .                                                                                                        |
| <code>OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED</code> | <code>OptixBuildInputOpacityMicromap::indexBuffer</code> provides a per triangle array of predefined indices and/or indices into <code>OptixBuildInputOpacityMicromap::opacityMicromapArray</code> . See <code>OptixBuildInputOpacityMicromap::indexBuffer</code> for more details. |

#### 5.14.4.29 OptixOpacityMicromapFlags

enum `OptixOpacityMicromapFlags`

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

Enumerator

|                                                            |                                                                                                   |
|------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| <code>OPTIX_OPACITY_MICROMAP_FLAG_NONE</code>              |                                                                                                   |
| <code>OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE</code> | This flag is mutually exclusive with <code>OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD</code> . |
| <code>OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD</code> | This flag is mutually exclusive with <code>OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE</code> . |

### 5.14.4.30 OptixOpacityMicromapFormat

enum `OptixOpacityMicromapFormat`

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

Enumerator

|                                                    |                                                                      |
|----------------------------------------------------|----------------------------------------------------------------------|
| <code>OPTIX_OPACITY_MICROMAP_FORMAT_NONE</code>    | invalid format                                                       |
| <code>OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE</code> | 0: Transparent, 1: Opaque                                            |
| <code>OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE</code> | 0: Transparent, 1: Opaque, 2: Unknown-Transparent, 3: Unknown-Opaque |

### 5.14.4.31 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

|                                                              |
|--------------------------------------------------------------|
| <code>OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE</code>       |
| <code>OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ</code>       |
| <code>OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE</code>      |
| <code>OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE</code> |
| <code>OPTIX_PAYLOAD_SEMANTICS_CH_NONE</code>                 |
| <code>OPTIX_PAYLOAD_SEMANTICS_CH_READ</code>                 |
| <code>OPTIX_PAYLOAD_SEMANTICS_CH_WRITE</code>                |
| <code>OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE</code>           |
| <code>OPTIX_PAYLOAD_SEMANTICS_MS_NONE</code>                 |
| <code>OPTIX_PAYLOAD_SEMANTICS_MS_READ</code>                 |
| <code>OPTIX_PAYLOAD_SEMANTICS_MS_WRITE</code>                |
| <code>OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE</code>           |
| <code>OPTIX_PAYLOAD_SEMANTICS_AH_NONE</code>                 |
| <code>OPTIX_PAYLOAD_SEMANTICS_AH_READ</code>                 |
| <code>OPTIX_PAYLOAD_SEMANTICS_AH_WRITE</code>                |
| <code>OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE</code>           |
| <code>OPTIX_PAYLOAD_SEMANTICS_IS_NONE</code>                 |

Enumerator

|                                       |
|---------------------------------------|
| OPTIX_PAYLOAD_SEMANTICS_IS_READ       |
| OPTIX_PAYLOAD_SEMANTICS_IS_WRITE      |
| OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE |

#### 5.14.4.32 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    |  |

#### 5.14.4.33 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           |

### 5.14.4.34 OptixPrimitiveType

enum `OptixPrimitiveType`

Builtin primitive types.

Enumerator

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

### 5.14.4.35 OptixPrimitiveTypeFlags

enum `OptixPrimitiveTypeFlags`

Builtin flags may be bitwise combined.

See also `OptixPipelineCompileOptions::usesPrimitiveTypeFlags`

Enumerator

|                                                                 |                                                               |
|-----------------------------------------------------------------|---------------------------------------------------------------|
| <code>OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM</code>                  | Custom primitive.                                             |
| <code>OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE</code> | B-spline curve of degree 2 with circular cross-section.       |
| <code>OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE</code>     | B-spline curve of degree 3 with circular cross-section.       |
| <code>OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR</code>            | Piecewise linear curve with circular cross-section.           |
| <code>OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM</code>        | CatmullRom curve with circular cross-section.                 |
| <code>OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE</code>  | B-spline curve of degree 2 with oriented, flat cross-section. |
| <code>OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE</code>                  | Sphere.                                                       |
| <code>OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER</code>      | Bezier curve of degree 3 with circular cross-section.         |
| <code>OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE</code>                | Triangle.                                                     |

Enumerator

|                                                         |                                                 |
|---------------------------------------------------------|-------------------------------------------------|
| OPTIX_PRIMITIVE_TYPE_FLAGS_DISPLACED_MICROMESH_TRIANGLE | Triangle with an applied displacement micromap. |
|---------------------------------------------------------|-------------------------------------------------|

#### 5.14.4.36 OptixProgramGroupFlags

enum `OptixProgramGroupFlags`

Flags for program groups.

Enumerator

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

#### 5.14.4.37 OptixProgramGroupKind

enum `OptixProgramGroupKind`

Distinguishes different kinds of program groups.

Enumerator

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

#### 5.14.4.38 OptixQueryFunctionTableOptions

enum `OptixQueryFunctionTableOptions`

Options that can be passed to `optixQueryFunctionTable()`

Enumerator

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

### 5.14.4.39 OptixRayFlags

enum `OptixRayFlags`

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

See also `optixTrace()`

Enumerator

|                                                         |                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OPTIX_RAY_FLAG_NONE</code>                        | No change from the behavior configured for the individual AS.                                                                                                                                                                                                                                                                                                                                |
| <code>OPTIX_RAY_FLAG_DISABLE_ANYHIT</code>              | Disables anyhit programs for the ray. Overrides <code>OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT</code> . This flag is mutually exclusive with <code>OPTIX_RAY_FLAG_ENFORCE_ANYHIT</code> , <code>OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT</code> , <code>OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT</code> .                                                                                                    |
| <code>OPTIX_RAY_FLAG_ENFORCE_ANYHIT</code>              | Forces anyhit program execution for the ray. Overrides <code>OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT</code> as well as <code>OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT</code> . This flag is mutually exclusive with <code>OPTIX_RAY_FLAG_DISABLE_ANYHIT</code> , <code>OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT</code> , <code>OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT</code> .                                  |
| <code>OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT</code>      | Terminates the ray after the first hit and executes the closesthit program of that hit.                                                                                                                                                                                                                                                                                                      |
| <code>OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT</code>          | Disables closesthit programs for the ray, but still executes miss program in case of a miss.                                                                                                                                                                                                                                                                                                 |
| <code>OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES</code>  | Do not intersect triangle back faces (respects a possible face change due to instance flag <code>OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING</code> ). This flag is mutually exclusive with <code>OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES</code> .                                                                                                                                           |
| <code>OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES</code> | Do not intersect triangle front faces (respects a possible face change due to instance flag <code>OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING</code> ). This flag is mutually exclusive with <code>OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES</code> .                                                                                                                                           |
| <code>OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT</code>        | Do not intersect geometry which disables anyhit programs (due to setting geometry flag <code>OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT</code> or instance flag <code>OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT</code> ). This flag is mutually exclusive with <code>OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT</code> , <code>OPTIX_RAY_FLAG_ENFORCE_ANYHIT</code> , <code>OPTIX_RAY_FLAG_DISABLE_ANYHIT</code> . |

## Enumerator

|                                               |                                                                                                                                                                                                                                                                                                                                           |
|-----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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.                                                                                                                                                                                                                                                  |

## 5.14.4.40 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        |

## Enumerator

|                                             |
|---------------------------------------------|
| 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    |
| 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                         |

## 5.14.4.41 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                   |

## 5.14.4.42 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>              |

## 5.14.4.43 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. |

## 5.14.4.44 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> |

Enumerator

|                                             |                                                                        |
|---------------------------------------------|------------------------------------------------------------------------|
| OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM | SRT motion transform. See also <a href="#">OptixSRTMotionTransform</a> |
|---------------------------------------------|------------------------------------------------------------------------|

#### 5.14.4.45 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 |                                           |
| OPTIX_VERTEX_FORMAT_SNORM16_2 |                                           |

## 6 Namespace Documentation

### 6.1 optix\_impl Namespace Reference

#### Functions

- 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)`
- `OPTIX_MICROMAP_INLINE_FUNC float __uint_as_float (unsigned int x)`
- `OPTIX_MICROMAP_INLINE_FUNC unsigned int extractEvenBits (unsigned int x)`
- `OPTIX_MICROMAP_INLINE_FUNC unsigned int prefixEor (unsigned int x)`
- `OPTIX_MICROMAP_INLINE_FUNC void index2dbary (unsigned int index, unsigned int &u, unsigned int &v, unsigned int &w)`
- `OPTIX_MICROMAP_INLINE_FUNC void micro2bary (unsigned int index, unsigned int subdivisionLevel, float2 &bary0, float2 &bary1, float2 &bary2)`
- `OPTIX_MICROMAP_INLINE_FUNC float2 base2micro (const float2 &baseBarycentrics, const float2 microVertexBaseBarycentrics[3])`

## 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 optixGetInterpolatedTransformation() [1/2]

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

### 6.1.1.3 optixGetInterpolatedTransformation() [2/2]

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

#### 6.1.1.4 optixGetInterpolatedTransformationFromHandle()

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

#### 6.1.1.5 optixGetMatrixFromSrt()

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

#### 6.1.1.6 optixGetObjectToWorldTransformMatrix()

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

#### 6.1.1.7 optixGetWorldToObjectTransformMatrix()

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

#### 6.1.1.8 optixInvertMatrix()

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

#### 6.1.1.9 optixLdg()

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

## 6.1.1.10 optixLoadInterpolatedMatrixKey()

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

## 6.1.1.11 optixLoadInterpolatedSrtKey()

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

## 6.1.1.12 optixLoadReadOnlyAlign16()

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

## 6.1.1.13 optixMulFloat4()

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

## 6.1.1.14 optixMultiplyRowMatrix()

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

## 6.1.1.15 optixResolveMotionKey()

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

```
const float globalt) [static]
```

#### 6.1.1.16 optixTransformNormal()

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

#### 6.1.1.17 optixTransformPoint()

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

#### 6.1.1.18 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` parameter 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 =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](#)

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` | `FULLY_OPAQUE` | `FULLY_UNKNOWN_TRANSPARENT` | `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 OptixBuiltinISOptions::buildFlags`

Build flags, see `OptixBuildFlags`.

7.13.2.2 builtinISModuleType

`OptixPrimitiveType OptixBuiltinISOptions::builtinISModuleType`

7.13.2.3 curveEndcapFlags

`unsigned int OptixBuiltinISOptions::curveEndcapFlags`

End cap properties of curves, see `OptixCurveEndcapFlags`, 0 for non-curve types.

7.13.2.4 usesMotionBlur

`int OptixBuiltinISOptions::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`
- `OptixDenoiserAlphaMode` `denoiseAlpha`

7.16.1 Detailed Description

Options used by the denoiser.

See also `optixDenoiserCreate()`

7.16.2 Member Data Documentation

7.16.2.1 denoiseAlpha

`OptixDenoiserAlphaMode OptixDenoiserOptions::denoiseAlpha`

alpha denoise mode

7.16.2.2 guideAlbedo

`unsigned int OptixDenoiserOptions::guideAlbedo`

7.16.2.3 guideNormal

`unsigned int OptixDenoiserOptions::guideNormal`

7.17 OptixDenoiserParams Struct Reference

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

Public Attributes

- `CUdeviceptr hdrIntensity`
- `float blendFactor`
- `CUdeviceptr hdrAverageColor`
- `unsigned int temporalModeUsePreviousLayers`

7.17.1 Detailed Description

Various parameters used by the denoiser.

See also `optixDenoiserInvoke()`

`optixDenoiserComputeIntensity()`

`optixDenoiserComputeAverageColor()`

7.17.2 Member Data Documentation

7.17.2.1 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.

7.17.2.2 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. if set to null, average log color will be calculated automatically. See `hdrIntensity` for tiling, this also applies

here.

7.17.2.3 hdrIntensity

`CUdeviceptr OptixDenoiserParams::hdrIntensity`

average log intensity of input image (default null pointer). points to a single float. if set to null, autoexposure will be calculated automatically for the input image. Should be set to average log intensity of the entire image at least if tiling is used to get consistent autoexposure for all tiles.

7.17.2.4 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 values input data (the displacement blocks).

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 displacement values 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^{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^{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^{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 OptixBuiltinISOOptions *builtinISOOptions, 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)`
- `OptixResult(* optixPlaceholder001)(OptixDeviceContext context)`
- `OptixResult(* optixPlaceholder002)(OptixDeviceContext context)`

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 `optixPlaceholder001`

`OptixResult(* OptixFunctionTable::optixPlaceholder001) (OptixDeviceContext context)`

See `optixConvertPointerToTraversableHandle()`.

7.24.2.45 `optixPlaceholder002`

`OptixResult(* OptixFunctionTable::optixPlaceholder002) (OptixDeviceContext context)`

See `optixConvertPointerToTraversableHandle()`.

7.24.2.46 `optixProgramGroupCreate`

`OptixResult(* OptixFunctionTable::optixProgramGroupCreate) (OptixDeviceContext context, const OptixProgramGroupDesc *programDescriptions, unsigned int numProgramGroups, const OptixProgramGroupOptions *options, char *logString, size_t *logStringSize, OptixProgramGroup *programGroups)`

See `optixProgramGroupCreate()`.

7.24.2.47 `optixProgramGroupDestroy`

`OptixResult(* OptixFunctionTable::optixProgramGroupDestroy) (OptixProgramGroup programGroup)`

See `optixProgramGroupDestroy()`.

7.24.2.48 `optixProgramGroupGetStackSize`

`OptixResult(* OptixFunctionTable::optixProgramGroupGetStackSize) (OptixProgramGroup programGroup, OptixStackSizes *stackSizes, OptixPipeline pipeline)`

See `optixProgramGroupGetStackSize()`.

7.24.2.49 `optixSbtRecordPackHeader`

`OptixResult(* OptixFunctionTable::optixSbtRecordPackHeader) (OptixProgramGroup programGroup, void *sbtRecordHeaderHostPointer)`

See `optixConvertPointerToTraversableHandle()`.

7.24.2.50 `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`
- `const 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

const 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^{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^{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^{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

- `const OptixPayloadType * payloadType`

### 7.43.1 Detailed Description

Program group options.

See also `optixProgramGroupCreate()`

### 7.43.2 Member Data Documentation

#### 7.43.2.1 payloadType

`const 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 OptixStackSizes 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 optixTraverse (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)`
- `template<typename... Payload>`  
`static __forceinline__ __device__ void optixTraverse (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 optixReorder (unsigned int coherenceHint, unsigned int numCoherenceHintBits)`
- `static __forceinline__ __device__ void optixReorder ()`
- `template<typename... Payload>`  
`static __forceinline__ __device__ void optixInvoke (OptixPayloadTypeID type, Payload &... payload)`
- `template<typename... Payload>`  
`static __forceinline__ __device__ void optixInvoke (Payload &... payload)`

- `template<typename... RegAttributes>`  
`static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int sbtOffset, unsigned int sbtStride, unsigned int instIdx, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)`
- `template<typename... RegAttributes>`  
`static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int sbtOffset, unsigned int sbtStride, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)`
- `template<typename... RegAttributes>`  
`static __forceinline__ __device__ void optixMakeHitObjectWithRecord (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int sbtRecordIndex, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)`
- `static __forceinline__ __device__ void optixMakeMissHitObject (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime)`
- `static __forceinline__ __device__ void optixMakeNopHitObject ()`
- `static __forceinline__ __device__ bool optixHitObjectIsHit ()`
- `static __forceinline__ __device__ bool optixHitObjectIsMiss ()`
- `static __forceinline__ __device__ bool optixHitObjectIsNop ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize ()`
- `static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle (unsigned int index)`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind ()`
- `static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin ()`
- `static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection ()`
- `static __forceinline__ __device__ float optixHitObjectGetRayTmin ()`
- `static __forceinline__ __device__ float optixHitObjectGetRayTmax ()`
- `static __forceinline__ __device__ float optixHitObjectGetRayTime ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7 ()`
- `static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex ()`
- `static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer ()`
- `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__ 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 types)`
- 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 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 `__device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle (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__ 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 optixGetExceptionLineInfo()

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

### 8.1.3.25 optixGetGASMotionStepCount()

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

### 8.1.3.26 optixGetGASMotionTimeBegin()

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

### 8.1.3.27 optixGetGASMotionTimeEnd()

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

## 8.1.3.28 optixGetGASPointerFromHandle()

```
static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle (
 OptixTraversableHandle handle) [static]
```

## 8.1.3.29 optixGetGASTraversableHandle()

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

## 8.1.3.30 optixGetHitKind()

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

## 8.1.3.31 optixGetInstanceChildFromHandle()

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

## 8.1.3.32 optixGetInstanceId()

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

## 8.1.3.33 optixGetInstanceIdFromHandle()

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

## 8.1.3.34 optixGetInstanceIndex()

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

## 8.1.3.35 optixGetInstanceInverseTransformFromHandle()

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

## 8.1.3.36 optixGetInstanceTransformFromHandle()

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

## 8.1.3.37 optixGetInstanceTraversableFromIAS()

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

## 8.1.3.38 optixGetLaunchDimensions()

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

## 8.1.3.39 optixGetLaunchIndex()

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

## 8.1.3.40 optixGetLinearCurveVertexData()

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

## 8.1.3.41 optixGetMatrixMotionTransformFromHandle()

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

## 8.1.3.42 optixGetMicroTriangleBarycentricsData()

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

## 8.1.3.43 optixGetMicroTriangleVertexData()

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

## 8.1.3.44 optixGetObjectRayDirection()

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

## 8.1.3.45 optixGetObjectRayOrigin()

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

## 8.1.3.46 optixGetObjectToWorldTransformMatrix()

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

## 8.1.3.47 optixGetPayload\_0()

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

### 8.1.3.48 optixGetPayload\_1()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_1 ( ) *[static]*

### 8.1.3.49 optixGetPayload\_10()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_10 ( ) *[static]*

### 8.1.3.50 optixGetPayload\_11()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_11 ( ) *[static]*

### 8.1.3.51 optixGetPayload\_12()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_12 ( ) *[static]*

### 8.1.3.52 optixGetPayload\_13()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_13 ( ) *[static]*

### 8.1.3.53 optixGetPayload\_14()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_14 ( ) *[static]*

### 8.1.3.54 optixGetPayload\_15()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_15 ( ) *[static]*

### 8.1.3.55 optixGetPayload\_16()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_16 ( ) *[static]*

### 8.1.3.56 optixGetPayload\_17()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_17 ( ) *[static]*

### 8.1.3.57 optixGetPayload\_18()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_18 ( ) *[static]*

### 8.1.3.58 optixGetPayload\_19()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_19 ( ) *[static]*

### 8.1.3.59 optixGetPayload\_2()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_2 ( ) *[static]*

### 8.1.3.60 optixGetPayload\_20()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_20 ( ) *[static]*

### 8.1.3.61 optixGetPayload\_21()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_21 ( ) *[static]*

### 8.1.3.62 optixGetPayload\_22()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_22 ( ) *[static]*

### 8.1.3.63 optixGetPayload\_23()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_23 ( ) *[static]*

### 8.1.3.64 optixGetPayload\_24()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_24 ( ) *[static]*

### 8.1.3.65 optixGetPayload\_25()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_25 ( ) *[static]*

### 8.1.3.66 optixGetPayload\_26()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_26 ( ) *[static]*

### 8.1.3.67 optixGetPayload\_27()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_27 ( ) *[static]*

### 8.1.3.68 optixGetPayload\_28()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_28 ( ) *[static]*

### 8.1.3.69 optixGetPayload\_29()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_29 ( ) *[static]*

### 8.1.3.70 optixGetPayload\_3()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_3 ( ) *[static]*

### 8.1.3.71 optixGetPayload\_30()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_30 ( ) *[static]*

### 8.1.3.72 optixGetPayload\_31()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_31 ( ) *[static]*

### 8.1.3.73 optixGetPayload\_4()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_4 ( ) *[static]*

### 8.1.3.74 optixGetPayload\_5()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_5 ( ) *[static]*

### 8.1.3.75 optixGetPayload\_6()

static \_\_forceinline\_\_ \_\_device\_\_ unsigned int optixGetPayload\_6 ( ) *[static]*

## 8.1.3.76 optixGetPayload\_7()

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

## 8.1.3.77 optixGetPayload\_8()

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

## 8.1.3.78 optixGetPayload\_9()

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

## 8.1.3.79 optixGetPrimitiveIndex()

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

## 8.1.3.80 optixGetPrimitiveType() [1/2]

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

## 8.1.3.81 optixGetPrimitiveType() [2/2]

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

## 8.1.3.82 optixGetQuadraticBSplineVertexData()

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

## 8.1.3.83 optixGetRayFlags()

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

## 8.1.3.84 optixGetRayTime()

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

## 8.1.3.85 optixGetRayTmax()

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

## 8.1.3.86 optixGetRayTmin()

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

## 8.1.3.87 optixGetRayVisibilityMask()

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

*[static]*

### 8.1.3.88 optixGetRibbonNormal()

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

### 8.1.3.89 optixGetRibbonParameters()

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

### 8.1.3.90 optixGetRibbonVertexData()

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

### 8.1.3.91 optixGetSbtDataPointer()

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

### 8.1.3.92 optixGetSbtGASIndex()

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

### 8.1.3.93 optixGetSphereData()

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

### 8.1.3.94 optixGetSRTMotionTransformFromHandle()

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

## 8.1.3.95 optixGetStaticTransformFromHandle()

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

## 8.1.3.96 optixGetTransformListHandle()

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

## 8.1.3.97 optixGetTransformListSize()

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

## 8.1.3.98 optixGetTransformTypeFromHandle()

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

## 8.1.3.99 optixGetTriangleBarycentrics()

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

## 8.1.3.100 optixGetTriangleVertexData()

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

## 8.1.3.101 optixGetWorldRayDirection()

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

## 8.1.3.102 optixGetWorldRayOrigin()

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

## 8.1.3.103 optixGetWorldToObjectTransformMatrix()

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

#### 8.1.3.104 optixHitObjectGetAttribute\_0()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0
() [static]
```

#### 8.1.3.105 optixHitObjectGetAttribute\_1()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1
() [static]
```

#### 8.1.3.106 optixHitObjectGetAttribute\_2()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2
() [static]
```

#### 8.1.3.107 optixHitObjectGetAttribute\_3()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3
() [static]
```

#### 8.1.3.108 optixHitObjectGetAttribute\_4()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4
() [static]
```

#### 8.1.3.109 optixHitObjectGetAttribute\_5()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5
() [static]
```

#### 8.1.3.110 optixHitObjectGetAttribute\_6()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6
() [static]
```

#### 8.1.3.111 optixHitObjectGetAttribute\_7()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7
() [static]
```

#### 8.1.3.112 optixHitObjectGetHitKind()

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

#### 8.1.3.113 optixHitObjectGetInstanceId()

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

#### 8.1.3.114 optixHitObjectGetInstanceIndex()

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

### 8.1.3.115 optixHitObjectGetPrimitiveIndex()

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

### 8.1.3.116 optixHitObjectGetRayTime()

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

### 8.1.3.117 optixHitObjectGetRayTmax()

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

### 8.1.3.118 optixHitObjectGetRayTmin()

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

### 8.1.3.119 optixHitObjectGetSbtDataPointer()

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

### 8.1.3.120 optixHitObjectGetSbtGASIndex()

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

### 8.1.3.121 optixHitObjectGetSbtRecordIndex()

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

### 8.1.3.122 optixHitObjectGetTransformListHandle()

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

### 8.1.3.123 optixHitObjectGetTransformListSize()

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

### 8.1.3.124 optixHitObjectGetWorldRayDirection()

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

### 8.1.3.125 optixHitObjectGetWorldRayOrigin()

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

### 8.1.3.126 optixHitObjectIsHit()

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

## 8.1.3.127 optixHitObjectIsMiss()

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

## 8.1.3.128 optixHitObjectIsNop()

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

## 8.1.3.129 optixIgnoreIntersection()

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

## 8.1.3.130 optixInvoke() [1/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixInvoke (
 OptixPayloadTypeID type,
 Payload &... payload) [static]
```

## 8.1.3.131 optixInvoke() [2/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixInvoke (
 Payload &... payload) [static]
```

## 8.1.3.132 optixIsBackFaceHit() [1/2]

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

## 8.1.3.133 optixIsBackFaceHit() [2/2]

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

## 8.1.3.134 optixIsDisplacedMicromeshTriangleBackFaceHit()

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

## 8.1.3.135 optixIsDisplacedMicromeshTriangleFrontFaceHit()

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

## 8.1.3.136 optixIsDisplacedMicromeshTriangleHit()

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

## 8.1.3.137 optixIsFrontFaceHit() [1/2]

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

## 8.1.3.138 optixIsFrontFaceHit() [2/2]

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

## 8.1.3.139 optixIsTriangleBackFaceHit()

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

## 8.1.3.140 optixIsTriangleFrontFaceHit()

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

## 8.1.3.141 optixIsTriangleHit()

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

## 8.1.3.142 optixMakeHitObject() [1/2]

```
template<typename... RegAttributes>
static __forceinline__ __device__ void optixMakeHitObject (
 OptixTraversableHandle handle,
 float3 rayOrigin,
 float3 rayDirection,
 float tmin,
 float tmax,
 float rayTime,
 unsigned int sbtOffset,
 unsigned int sbtStride,
 unsigned int instIdx,
 const OptixTraversableHandle * transforms,
 unsigned int numTransforms,
 unsigned int sbtGASIdx,
 unsigned int primIdx,
 unsigned int hitKind,
 RegAttributes... regAttributes) [static]
```

## 8.1.3.143 optixMakeHitObject() [2/2]

```
template<typename... RegAttributes>
static __forceinline__ __device__ void optixMakeHitObject (
 OptixTraversableHandle handle,
 float3 rayOrigin,
 float3 rayDirection,
 float tmin,
 float tmax,
 float rayTime,
 unsigned int sbtOffset,
```

```

 unsigned int sbtStride,
 unsigned int instIdx,
 unsigned int sbtGASIdx,
 unsigned int primIdx,
 unsigned int hitKind,
 RegAttributes... regAttributes) [static]

```

#### 8.1.3.144 optixMakeHitObjectWithRecord()

```

template<typename... RegAttributes>
static __forceinline__ __device__ void optixMakeHitObjectWithRecord (
 OptixTraversableHandle handle,
 float3 rayOrigin,
 float3 rayDirection,
 float tmin,
 float tmax,
 float rayTime,
 unsigned int sbtRecordIndex,
 unsigned int instIdx,
 const OptixTraversableHandle * transforms,
 unsigned int numTransforms,
 unsigned int sbtGASIdx,
 unsigned int primIdx,
 unsigned int hitKind,
 RegAttributes... regAttributes) [static]

```

#### 8.1.3.145 optixMakeMissHitObject()

```

static __forceinline__ __device__ void optixMakeMissHitObject (
 unsigned int missSBTIndex,
 float3 rayOrigin,
 float3 rayDirection,
 float tmin,
 float tmax,
 float rayTime) [static]

```

#### 8.1.3.146 optixMakeNopHitObject()

```

static __forceinline__ __device__ void optixMakeNopHitObject () [static]

```

#### 8.1.3.147 optixReorder() [1/2]

```

static __forceinline__ __device__ void optixReorder () [static]

```

#### 8.1.3.148 optixReorder() [2/2]

```

static __forceinline__ __device__ void optixReorder (

```

```
 unsigned int coherenceHint,
 unsigned int numCoherenceHintBits) [static]
```

#### 8.1.3.149 optixReportIntersection() [1/9]

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

#### 8.1.3.150 optixReportIntersection() [2/9]

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

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

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

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

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

#### 8.1.3.153 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]
```

#### 8.1.3.154 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]
```

#### 8.1.3.155 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]
```

#### 8.1.3.156 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]
```

#### 8.1.3.157 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]
```

## 8.1.3.158 optixSetPayload\_0()

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

## 8.1.3.159 optixSetPayload\_1()

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

## 8.1.3.160 optixSetPayload\_10()

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

## 8.1.3.161 optixSetPayload\_11()

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

## 8.1.3.162 optixSetPayload\_12()

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

## 8.1.3.163 optixSetPayload\_13()

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

## 8.1.3.164 optixSetPayload\_14()

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

## 8.1.3.165 optixSetPayload\_15()

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

## 8.1.3.166 optixSetPayload\_16()

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

## 8.1.3.167 optixSetPayload\_17()

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

## 8.1.3.168 optixSetPayload\_18()

```
static __forceinline__ __device__ void optixSetPayload_18 (
```

unsigned int *p* ) *[static]*

#### 8.1.3.169 optixSetPayload\_19()

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

#### 8.1.3.170 optixSetPayload\_2()

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

#### 8.1.3.171 optixSetPayload\_20()

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

#### 8.1.3.172 optixSetPayload\_21()

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

#### 8.1.3.173 optixSetPayload\_22()

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

#### 8.1.3.174 optixSetPayload\_23()

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

#### 8.1.3.175 optixSetPayload\_24()

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

#### 8.1.3.176 optixSetPayload\_25()

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

#### 8.1.3.177 optixSetPayload\_26()

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

#### 8.1.3.178 optixSetPayload\_27()

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

### 8.1.3.179 optixSetPayload\_28()

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

### 8.1.3.180 optixSetPayload\_29()

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

### 8.1.3.181 optixSetPayload\_3()

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

### 8.1.3.182 optixSetPayload\_30()

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

### 8.1.3.183 optixSetPayload\_31()

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

### 8.1.3.184 optixSetPayload\_4()

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

### 8.1.3.185 optixSetPayload\_5()

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

### 8.1.3.186 optixSetPayload\_6()

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

### 8.1.3.187 optixSetPayload\_7()

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

### 8.1.3.188 optixSetPayload\_8()

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

### 8.1.3.189 optixSetPayload\_9()

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

```
 unsigned int p) [static]
```

#### 8.1.3.190 optixSetPayloadTypes()

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

#### 8.1.3.191 optixTerminateRay()

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

#### 8.1.3.192 optixTexFootprint2D()

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

#### 8.1.3.193 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]
```

#### 8.1.3.194 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]
```

#### 8.1.3.195 optixThrowException() [1/9]

```
static __forceinline__ __device__ void optixThrowException (
```

```
int exceptionCode) [static]
```

#### 8.1.3.196 optixThrowException() [2/9]

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

#### 8.1.3.197 optixThrowException() [3/9]

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

#### 8.1.3.198 optixThrowException() [4/9]

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

#### 8.1.3.199 optixThrowException() [5/9]

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

#### 8.1.3.200 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]
```

#### 8.1.3.201 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]
```

#### 8.1.3.202 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]
```

#### 8.1.3.203 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]
```

#### 8.1.3.204 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]
```

### 8.1.3.205 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]
```

### 8.1.3.206 optixTransformNormalFromObjectToWorldSpace()

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

### 8.1.3.207 optixTransformNormalFromWorldToObjectSpace()

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

### 8.1.3.208 optixTransformPointFromObjectToWorldSpace()

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

### 8.1.3.209 optixTransformPointFromWorldToObjectSpace()

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

### 8.1.3.210 optixTransformVectorFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixTransformVectorFromObjectToWorldSpace (
```

```
float3 vec) [static]
```

### 8.1.3.211 optixTransformVectorFromWorldToObjectSpace()

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

### 8.1.3.212 optixTraverse() [1/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixTraverse (
 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]
```

### 8.1.3.213 optixTraverse() [2/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixTraverse (
 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]
```

### 8.1.3.214 optixUndefinedValue()

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

## 8.2 optix\_device\_impl.h

Go to the documentation of this file.

```

1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
4 *
5 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
6 * property and proprietary rights in and to this material, related
7 * documentation and any modifications thereto. Any use, reproduction,
8 * disclosure or distribution of this material and related documentation
9 * without an express license agreement from NVIDIA CORPORATION or
10 * its affiliates is strictly prohibited.
11 */
12 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
13 #error("optix_device_impl.h is an internal header file and must not be used directly. Please use
14 optix_device.h or optix.h instead.")
15 #endif
16
17 #ifndef OPTIX_DEVICE_IMPL_H
18 #define OPTIX_DEVICE_IMPL_H
19
20 #include "internal/optix_device_impl_transformations.h"
21
22 #ifndef __CUDACC_RTC__
23 #include <initializer_list>
24 #include <type_traits>
25 #endif
26
27 namespace optix_internal {
28 template <typename...>
29 struct TypePack{};
30 } // namespace optix_internal
31
32 template <typename... Payload>
33 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
34 float3 rayOrigin,
35 float3 rayDirection,
36 float tmin,
37 float tmax,
38 float rayTime,
39 OptixVisibilityMask visibilityMask,
40 unsigned int rayFlags,
41 unsigned int SBTOffset,
42 unsigned int SBTStride,
43 unsigned int missSBTIndex,
44 Payload&... payload)
45 {
46 static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
47 // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
48 // TypePack 1 unsigned int T0 T1 T2 ... Tn-1 Tn
49 // TypePack 2 T0 T1 T2 T3 ... Tn unsigned int
50 #ifndef __CUDACC_RTC__
51 static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
52 optix_internal::TypePack<Payload..., unsigned int>::value,
53 "All payload parameters need to be unsigned int.");
54 #endif
55
56 OptixPayloadTypeID type = OPTIX_PAYLOAD_TYPE_DEFAULT;
57 float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
58 float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
59 unsigned int p[33] = { 0, payload... };
60 int payloadSize = (int)sizeof...(Payload);
61 asm volatile(
62 "call"
63);
64 }
65
66 #endif
67
68 #endif
69
70 (%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,%

```

```

70 "29,%30,%31),"
71 "_optix_trace_typed_32,"
72
73 "(%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,%
74 "59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
75 : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
76 "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
77 "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
78 "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
79 "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
80 : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
81 "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBToffset), "r"(SBTstride),
82 "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
83 "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
84 "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
85 "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
86 "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
87);
88 unsigned int index = 1;
89 (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
90 }
91 template <typename... Payload>
92 static __forceinline__ __device__ void optixTraverse(OptixTraversableHandle handle,
93 float3 rayOrigin,
94 float3 rayDirection,
95 float tmin,
96 float tmax,
97 float rayTime,
98 OptixVisibilityMask visibilityMask,
99 unsigned int rayFlags,
100 unsigned int SBToffset,
101 unsigned int SBTstride,
102 unsigned int missSBTIndex,
103 Payload&... payload)
104 {
105 static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
106 // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
107 // TypePack 1 unsigned int T0 T1 T2 ... Tn-1 Tn
108 // TypePack 2 T0 T1 T2 T3 ... Tn unsigned int
109 #ifndef __CUDA_RTC__
110 static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
111 optix_internal::TypePack<Payload..., unsigned int>::value,
112 "All payload parameters need to be unsigned int.");
113 #endif
114 OptixPayloadTypeID type = OPTIX_PAYLOAD_TYPE_DEFAULT;
115 float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
116 float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
117 unsigned int p[33] = {0, payload...};
118 int payloadSize = (int)sizeof...(Payload);
119 asm volatile(
120 "call"
121
122 "(%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,%
123 "29,%30,%31),"
124 "_optix_hitobject_traverse,"
125 "(%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,%
126 "59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
127 : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
128 "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
129 "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
130 "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
131 "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
132 : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
133 "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBToffset), "r"(SBTstride),

```

```

133 "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
134 "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
135 "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
136 "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
137 "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
138);
139 unsigned int index = 1;
140 (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
141 }
142
143 template <typename... Payload>
144 static __forceinline__ __device__ void optixTrace(OptixPayloadTypeID type,
145 OptixTraversableHandle handle,
146 float3 rayOrigin,
147 float3 rayDirection,
148 float tmin,
149 float tmax,
150 float rayTime,
151 OptixVisibilityMask visibilityMask,
152 unsigned int rayFlags,
153 unsigned int SBTOffset,
154 unsigned int SBTStride,
155 unsigned int missSBTIndex,
156 Payload&... payload)
157 {
158 // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
159 // TypePack 1 unsigned int T0 T1 T2 ... Tn-1 Tn
160 // TypePack 2 T0 T1 T2 T3 ... Tn unsigned int
161 static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
162 #ifndef __CUDA_RTC__
163 static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int>::value,
164 "All payload parameters need to be unsigned int.");
165 #endif
166
167 float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
168 float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
169 unsigned int p[33] = {0, payload...};
170 int payloadSize = (int)sizeof...(Payload);
171
172 asm volatile(
173 "call"
174
175 "(%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),",
176 "_optix_trace_typed_32,",
177
178 "(%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);",
179 : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
180 "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
181 "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
182 "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
183 "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
184 : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
185 "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBTOffset), "r"(SBTStride),
186 "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
187 "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
188 "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]), "r"(p[21]),
189 "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]), "r"(p[28]),
190 "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
191);
192 unsigned int index = 1;
193 (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
194 }
195
196 template <typename... Payload>

```

```

197 static __forceinline__ __device__ void optixTraverse(OptixPayloadTypeID type,
198 OptixTraversableHandle handle,
199 float3 rayOrigin,
200 float3 rayDirection,
201 float tmin,
202 float tmax,
203 float rayTime,
204 OptixVisibilityMask visibilityMask,
205 unsigned int rayFlags,
206 unsigned int SBTOffset,
207 unsigned int SBTstride,
208 unsigned int missSBTIndex,
209 Payload&... payload)
210 {
211 // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
212 // TypePack 1 unsigned int T0 T1 T2 ... Tn-1 Tn
213 // TypePack 2 T0 T1 T2 T3 ... Tn unsigned int
214 static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
215 #ifndef __CUDAACC_RTC__
216 static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
217 optix_internal::TypePack<Payload..., unsigned int>::value,
218 "All payload parameters need to be unsigned int.");
219 #endif
220 float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
221 float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
222 unsigned int p[33] = {0, payload...};
223 int payloadSize = (int)sizeof...(Payload);
224 asm volatile(
225 "call"
226
227 "(%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), "
228 "_optix_hitobject_traverse,"
229
230 "(%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); "
231 : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
232 "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
233 "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
234 "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
235 "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
236 : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
237 "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBTOffset), "r"(SBTstride),
238 "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
239 "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
240 "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
241 "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
242 "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
243 :);
244 unsigned int index = 1;
245 (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
246 }
247
248 static __forceinline__ __device__ void optixReorder(unsigned int coherenceHint, unsigned int
249 numCoherenceHintBits)
250 {
251 asm volatile(
252 "call"
253 "(), "
254 "_optix_hitobject_reorder,"
255 "(%0,%1);"
256 : "r"(coherenceHint), "r"(numCoherenceHintBits)
257 :);
258 }
259

```

```

260 static __forceinline__ __device__ void optixReorder()
261 {
262 unsigned int coherenceHint = 0;
263 unsigned int numCoherenceHintBits = 0;
264 asm volatile(
265 "call"
266 "(",
267 "_optix_hitobject_reorder,"
268 "(%0,%1);"
269 :
270 : "r"(coherenceHint), "r"(numCoherenceHintBits)
271 :);
272 }
273
274 template <typename... Payload>
275 static __forceinline__ __device__ void optixInvoke(OptixPayloadTypeID type, Payload&... payload)
276 {
277 // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
278 // TypePack 1 unsigned int T0 T1 T2 ... Tn-1 Tn
279 // TypePack 2 T0 T1 T2 T3 ... Tn unsigned int
280 static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
281 #ifndef __CUDACC_RTC__
282 static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int>::value,
283 "All payload parameters need to be unsigned int.");
284 #endif
285
286 unsigned int p[33] = {0, payload...};
287 int payloadSize = (int)sizeof...(Payload);
288
289 asm volatile(
290 "call"
291 "(%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),"
292 "_optix_hitobject_invoke,"
293 "(%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);"
294 : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
295 "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
296 "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
297 "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
298 "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
299 : "r"(type), "r"(payloadSize), "r"(p[1]), "r"(p[2]),
300 "r"(p[3]), "r"(p[4]), "r"(p[5]), "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]),
301 "r"(p[11]), "r"(p[12]), "r"(p[13]), "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]),
302 "r"(p[18]), "r"(p[19]), "r"(p[20]), "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]),
303 "r"(p[25]), "r"(p[26]), "r"(p[27]), "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
304 :);
305
306 unsigned int index = 1;
307 (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
308 }
309
310 template <typename... Payload>
311 static __forceinline__ __device__ void optixInvoke(Payload&... payload)
312 {
313 // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
314 // TypePack 1 unsigned int T0 T1 T2 ... Tn-1 Tn
315 // TypePack 2 T0 T1 T2 T3 ... Tn unsigned int
316 static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
317 #ifndef __CUDACC_RTC__
318 static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int>::value,
319 "All payload parameters need to be unsigned int.");
320 #endif
321 }
322 #endif

```

```

323
324 OptixPayloadTypeID type = OPTIX_PAYLOAD_TYPE_DEFAULT;
325 unsigned int p[33] = {0, payload...};
326 int payloadSize = (int)sizeof...(Payload);
327
328 asm volatile(
329 "call"
330
331 "(%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),"
332 "_optix_hitobject_invoke,"
333
334 "(%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);"
335 : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
336 "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
337 "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
338 "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
339 "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
340 : "r"(type), "r"(payloadSize), "r"(p[1]), "r"(p[2]),
341 "r"(p[3]), "r"(p[4]), "r"(p[5]), "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]),
342 "r"(p[11]), "r"(p[12]), "r"(p[13]), "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]),
343 "r"(p[18]), "r"(p[19]), "r"(p[20]), "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]),
344 "r"(p[25]), "r"(p[26]), "r"(p[27]), "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
345 :);
346
347 unsigned int index = 1;
348 (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
349 }
350
351 template <typename... RegAttributes>
352 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle handle,
353 float3 rayOrigin,
354 float3 rayDirection,
355 float tmin,
356 float tmax,
357 float rayTime,
358 unsigned int sbtOffset,
359 unsigned int sbtStride,
360 unsigned int instIdx,
361 unsigned int sbtGASIdx,
362 unsigned int primIdx,
363 unsigned int hitKind,
364 RegAttributes... regAttributes)
365 {
366 // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
367 // TypePack 1 unsigned int T0 T1 T2 ... Tn-1 Tn
368 // TypePack 2 T0 T1 T2 T3 ... Tn unsigned int
369 static_assert(sizeof...(RegAttributes) <= 8, "Only up to 8 register attribute values are allowed.");
370 #ifndef __CUDA_RTC__
371 static_assert(
372 std::is_same<optix_internal::TypePack<unsigned int, RegAttributes...>,
373 optix_internal::TypePack<RegAttributes..., unsigned int>::value,
374 "All register attribute parameters need to be unsigned int.");
375 #endif
376 float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
377 float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
378 unsigned int a[9] = {0, regAttributes...};
379 int attrSize = (int)sizeof...(RegAttributes);
380
381 OptixTraversableHandle* transforms = nullptr;
382 unsigned int numTransforms = 0;
383
384 asm volatile(
385 "call"
386 "(),"

```

```

387 "_optix_hitobject_make_hit,"
388
389 "(%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);"
390 : "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin), "f"(tmax),
391 "f"(rayTime), "r"(sbtOffset), "r"(sbtStride), "r"(instIdx), "l"(transforms),
392 "r"(numTransforms),
393 "r"(sbtGASIdx), "r"(primIdx), "r"(hitKind), "r"(attrSize), "r"(a[1]), "r"(a[2]), "r"(a[3]),
394 "r"(a[4]), "r"(a[5]), "r"(a[6]), "r"(a[7]), "r"(a[8])
395);
396 }
397 template <typename... RegAttributes>
398 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle handle,
399 float3 rayOrigin,
400 float3 rayDirection,
401 float tmin,
402 float tmax,
403 float rayTime,
404 unsigned int sbtOffset,
405 unsigned int sbtStride,
406 unsigned int instIdx,
407 const OptixTraversableHandle* transforms,
408 unsigned int numTransforms,
409 unsigned int sbtGASIdx,
410 unsigned int primIdx,
411 unsigned int hitKind,
412 RegAttributes... regAttributes)
413 {
414 // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
415 // TypePack 1 unsigned int T0 T1 T2 ... Tn-1 Tn
416 // TypePack 2 T0 T1 T2 T3 ... Tn unsigned int
417 static_assert(sizeof...(RegAttributes) <= 8, "Only up to 8 register attribute values are allowed.");
418 #ifndef __CUDA_RTC__
419 static_assert(
420 std::is_same<optix_internal::TypePack<unsigned int, RegAttributes...>,
421 optix_internal::TypePack<RegAttributes..., unsigned int>::value,
422 "All register attribute parameters need to be unsigned int.");
423 #endif
424 float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
425 float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
426 unsigned int a[9] = {0, regAttributes...};
427 int attrSize = (int)sizeof...(RegAttributes);
428
429 asm volatile(
430 "call"
431 "(),"
432 "_optix_hitobject_make_hit,"
433
434 "(%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);"
435 : "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin), "f"(tmax),
436 "f"(rayTime), "r"(sbtOffset), "r"(sbtStride), "r"(instIdx), "l"(transforms),
437 "r"(numTransforms),
438 "r"(sbtGASIdx), "r"(primIdx), "r"(hitKind), "r"(attrSize), "r"(a[1]), "r"(a[2]), "r"(a[3]),
439 "r"(a[4]), "r"(a[5]), "r"(a[6]), "r"(a[7]), "r"(a[8])
440);
441
442 template <typename... RegAttributes>
443 static __forceinline__ __device__ void optixMakeHitObjectWithRecord(OptixTraversableHandle handle,
444 float3 rayOrigin,
445 float3 rayDirection,
446 float tmin,
447 float tmax,
448 float rayTime,

```

```

449 unsigned int sbtRecordIndex,
450 unsigned int instIdx,
451 const OptixTraversableHandle* transforms,
452 unsigned int numTransforms,
453 unsigned int sbtGASIdx,
454 unsigned int primIdx,
455 unsigned int hitKind,
456 RegAttributes... regAttributes)
457 {
458 // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
459 // TypePack 1 unsigned int T0 T1 T2 ... Tn-1 Tn
460 // TypePack 2 T0 T1 T2 T3 ... Tn unsigned int
461 static_assert(sizeof...(RegAttributes) <= 8, "Only up to 8 register attribute values are allowed.");
462 #ifndef __CUDACC_RTC__
463 static_assert(
464 std::is_same<optix_internal::TypePack<unsigned int, RegAttributes...>,
465 optix_internal::TypePack<RegAttributes..., unsigned int>::value,
466 "All register attribute parameters need to be unsigned int.");
467 #endif
468 float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
469 float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
470 unsigned int a[9] = {0, regAttributes...};
471 int attrSize = (int)sizeof...(RegAttributes);
472
473 asm volatile(
474 "call"
475 "(),"
476 "_optix_hitobject_make_hit_with_record,"
477
478 "(%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);"
479 :
480 : "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin), "f"(tmax),
481 "f"(rayTime), "r"(sbtRecordIndex), "r"(instIdx), "l"(transforms), "r"(numTransforms),
482 "r"(sbtGASIdx), "r"(primIdx), "r"(hitKind), "r"(attrSize), "r"(a[1]), "r"(a[2]), "r"(a[3]),
483 "r"(a[4]), "r"(a[5]), "r"(a[6]), "r"(a[7]), "r"(a[8])
484);
485
486 static __forceinline__ __device__ void optixMakeMissHitObject(unsigned int missSBTIndex,
487 float3 rayOrigin,
488 float3 rayDirection,
489 float tmin,
490 float tmax,
491 float rayTime)
492 {
493 float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
494 float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
495
496 asm volatile(
497 "call"
498 "(),"
499 "_optix_hitobject_make_miss,"
500 "(%0,%1,%2,%3,%4,%5,%6,%7,%8,%9);"
501 :
502 : "r"(missSBTIndex), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
503 "f"(tmax), "f"(rayTime)
504);
505 }
506
507 static __forceinline__ __device__ void optixMakeNopHitObject()
508 {
509 asm volatile(
510 "call"
511 "(),"
512 "_optix_hitobject_make_nop,"
513 "();"

```

```
514 :
515 :
516 :);
517 }
518
519 static __forceinline__ __device__ bool optixHitObjectIsHit()
520 {
521 unsigned int result;
522 asm volatile(
523 "call (%0), _optix_hitobject_is_hit,"
524 "();"
525 : "=r"(result)
526 :
527 :);
528 return result;
529 }
530
531 static __forceinline__ __device__ bool optixHitObjectIsMiss()
532 {
533 unsigned int result;
534 asm volatile(
535 "call (%0), _optix_hitobject_is_miss,"
536 "();"
537 : "=r"(result)
538 :
539 :);
540 return result;
541 }
542
543 static __forceinline__ __device__ bool optixHitObjectIsNop()
544 {
545 unsigned int result;
546 asm volatile(
547 "call (%0), _optix_hitobject_is_nop,"
548 "();"
549 : "=r"(result)
550 :
551 :);
552 return result;
553 }
554
555 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId()
556 {
557 unsigned int result;
558 asm volatile(
559 "call (%0), _optix_hitobject_get_instance_id,"
560 "();"
561 : "=r"(result)
562 :
563 :);
564 return result;
565 }
566
567 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex()
568 {
569 unsigned int result;
570 asm volatile(
571 "call (%0), _optix_hitobject_get_instance_idx,"
572 "();"
573 : "=r"(result)
574 :
575 :);
576 return result;
577 }
578
579 static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex()
580 {
```

```

581 unsigned int result;
582 asm volatile(
583 "call (%0), _optix_hitobject_get_primitive_idx,"
584 "();"
585 : "=r"(result)
586 :
587 :);
588 return result;
589 }
590
591 static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize()
592 {
593 unsigned int result;
594 asm volatile(
595 "call (%0), _optix_hitobject_get_transform_list_size,"
596 "();"
597 : "=r"(result)
598 :
599 :);
600 return result;
601 }
602
603 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle(unsigned
int index)
604 {
605 unsigned long long result;
606 asm volatile(
607 "call (%0), _optix_hitobject_get_transform_list_handle,"
608 "(%1);"
609 : "=l"(result)
610 : "r"(index)
611 :);
612 return result;
613 }
614
615 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex()
616 {
617 unsigned int result;
618 asm volatile(
619 "call (%0), _optix_hitobject_get_sbt_gas_idx,"
620 "();"
621 : "=r"(result)
622 :
623 :);
624 return result;
625 }
626
627 static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind()
628 {
629 unsigned int result;
630 asm volatile(
631 "call (%0), _optix_hitobject_get_hitkind,"
632 "();"
633 : "=r"(result)
634 :
635 :);
636 return result;
637 }
638
639 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin()
640 {
641 float x, y, z;
642 asm volatile(
643 "call (%0), _optix_hitobject_get_world_ray_origin_x,"
644 "();"
645 : "=f"(x)
646 :

```

```

647);
648 asm volatile(
649 "call (%0), _optix_hitobject_get_world_ray_origin_y,"
650 "();"
651 : "=f"(y)
652 :
653 :);
654 asm volatile(
655 "call (%0), _optix_hitobject_get_world_ray_origin_z,"
656 "();"
657 : "=f"(z)
658 :
659 :);
660 return make_float3(x, y, z);
661 }
662
663 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection()
664 {
665 float x, y, z;
666 asm volatile(
667 "call (%0), _optix_hitobject_get_world_ray_direction_x,"
668 "();"
669 : "=f"(x)
670 :
671 :);
672 asm volatile(
673 "call (%0), _optix_hitobject_get_world_ray_direction_y,"
674 "();"
675 : "=f"(y)
676 :
677 :);
678 asm volatile(
679 "call (%0), _optix_hitobject_get_world_ray_direction_z,"
680 "();"
681 : "=f"(z)
682 :
683 :);
684 return make_float3(x, y, z);
685 }
686
687 static __forceinline__ __device__ float optixHitObjectGetRayTmin()
688 {
689 float result;
690 asm volatile(
691 "call (%0), _optix_hitobject_get_ray_tmin,"
692 "();"
693 : "=f"(result)
694 :
695 :);
696 return result;
697 }
698
699 static __forceinline__ __device__ float optixHitObjectGetRayTmax()
700 {
701 float result;
702 asm volatile(
703 "call (%0), _optix_hitobject_get_ray_tmax,"
704 "();"
705 : "=f"(result)
706 :
707 :);
708 return result;
709 }
710
711 static __forceinline__ __device__ float optixHitObjectGetRayTime()
712 {
713 float result;

```

```
714 asm volatile(
715 "call (%0), _optix_hitobject_get_ray_time,"
716 "());"
717 : "=f"(result)
718 :
719 :);
720 return result;
721 }
722
723 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0()
724 {
725 unsigned int ret;
726 asm volatile(
727 "call (%0), _optix_hitobject_get_attribute,"
728 "(%1);"
729 : "=r"(ret)
730 : "r"(0)
731 :);
732 return ret;
733 }
734
735 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1()
736 {
737 unsigned int ret;
738 asm volatile(
739 "call (%0), _optix_hitobject_get_attribute,"
740 "(%1);"
741 : "=r"(ret)
742 : "r"(1)
743 :);
744 return ret;
745 }
746
747 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2()
748 {
749 unsigned int ret;
750 asm volatile(
751 "call (%0), _optix_hitobject_get_attribute,"
752 "(%1);"
753 : "=r"(ret)
754 : "r"(2)
755 :);
756 return ret;
757 }
758
759 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3()
760 {
761 unsigned int ret;
762 asm volatile(
763 "call (%0), _optix_hitobject_get_attribute,"
764 "(%1);"
765 : "=r"(ret)
766 : "r"(3)
767 :);
768 return ret;
769 }
770
771 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4()
772 {
773 unsigned int ret;
774 asm volatile(
775 "call (%0), _optix_hitobject_get_attribute,"
776 "(%1);"
777 : "=r"(ret)
778 : "r"(4)
779 :);
780 return ret;
781 }
```

```

781 }
782
783 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5()
784 {
785 unsigned int ret;
786 asm volatile(
787 "call (%0), _optix_hitobject_get_attribute,"
788 "(%1);"
789 : "=r"(ret)
790 : "r"(5)
791 :);
792 return ret;
793 }
794
795 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6()
796 {
797 unsigned int ret;
798 asm volatile(
799 "call (%0), _optix_hitobject_get_attribute,"
800 "(%1);"
801 : "=r"(ret)
802 : "r"(6)
803 :);
804 return ret;
805 }
806
807 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7()
808 {
809 unsigned int ret;
810 asm volatile(
811 "call (%0), _optix_hitobject_get_attribute,"
812 "(%1);"
813 : "=r"(ret)
814 : "r"(7)
815 :);
816 return ret;
817 }
818
819 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex()
820 {
821 unsigned int result;
822 asm volatile(
823 "call (%0), _optix_hitobject_get_sbt_record_index,"
824 "();"
825 : "=r"(result)
826 :
827 :);
828 return result;
829 }
830
831 static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer()
832 {
833 unsigned long long ptr;
834 asm volatile(
835 "call (%0), _optix_hitobject_get_sbt_data_pointer,"
836 "();"
837 : "=l"(ptr)
838 :
839 :);
840 return ptr;
841 }
842
843 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p)
844 {
845 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(0), "r"(p) :);
846 }
847

```

```
848 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p)
849 {
850 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(1), "r"(p) :);
851 }
852
853 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p)
854 {
855 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(2), "r"(p) :);
856 }
857
858 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p)
859 {
860 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(3), "r"(p) :);
861 }
862
863 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p)
864 {
865 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(4), "r"(p) :);
866 }
867
868 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p)
869 {
870 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(5), "r"(p) :);
871 }
872
873 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p)
874 {
875 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(6), "r"(p) :);
876 }
877
878 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p)
879 {
880 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(7), "r"(p) :);
881 }
882
883 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p)
884 {
885 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(8), "r"(p) :);
886 }
887
888 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p)
889 {
890 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(9), "r"(p) :);
891 }
892
893 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p)
894 {
895 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(10), "r"(p) :);
896 }
897
898 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p)
899 {
900 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(11), "r"(p) :);
901 }
902
903 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p)
904 {
905 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(12), "r"(p) :);
906 }
907
908 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p)
909 {
910 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(13), "r"(p) :);
911 }
912
913 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p)
914 {
```

```
915 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(14), "r"(p) :);
916 }
917
918 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p)
919 {
920 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(15), "r"(p) :);
921 }
922
923 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p)
924 {
925 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(16), "r"(p) :);
926 }
927
928 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p)
929 {
930 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(17), "r"(p) :);
931 }
932
933 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p)
934 {
935 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(18), "r"(p) :);
936 }
937
938 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p)
939 {
940 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(19), "r"(p) :);
941 }
942
943 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p)
944 {
945 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(20), "r"(p) :);
946 }
947
948 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p)
949 {
950 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(21), "r"(p) :);
951 }
952
953 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p)
954 {
955 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(22), "r"(p) :);
956 }
957
958 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p)
959 {
960 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(23), "r"(p) :);
961 }
962
963 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p)
964 {
965 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(24), "r"(p) :);
966 }
967
968 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p)
969 {
970 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(25), "r"(p) :);
971 }
972
973 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p)
974 {
975 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(26), "r"(p) :);
976 }
977
978 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p)
979 {
980 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(27), "r"(p) :);
981 }
```

```
982
983 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p)
984 {
985 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(28), "r"(p) :);
986 }
987
988 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p)
989 {
990 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(29), "r"(p) :);
991 }
992
993 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p)
994 {
995 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(30), "r"(p) :);
996 }
997
998 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p)
999 {
1000 asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(31), "r"(p) :);
1001 }
1002
1003 static __forceinline__ __device__ unsigned int optixGetPayload_0()
1004 {
1005 unsigned int result;
1006 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(0) :);
1007 return result;
1008 }
1009
1010 static __forceinline__ __device__ unsigned int optixGetPayload_1()
1011 {
1012 unsigned int result;
1013 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(1) :);
1014 return result;
1015 }
1016
1017 static __forceinline__ __device__ unsigned int optixGetPayload_2()
1018 {
1019 unsigned int result;
1020 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(2) :);
1021 return result;
1022 }
1023
1024 static __forceinline__ __device__ unsigned int optixGetPayload_3()
1025 {
1026 unsigned int result;
1027 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(3) :);
1028 return result;
1029 }
1030
1031 static __forceinline__ __device__ unsigned int optixGetPayload_4()
1032 {
1033 unsigned int result;
1034 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(4) :);
1035 return result;
1036 }
1037
1038 static __forceinline__ __device__ unsigned int optixGetPayload_5()
1039 {
1040 unsigned int result;
1041 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(5) :);
1042 return result;
1043 }
1044
1045 static __forceinline__ __device__ unsigned int optixGetPayload_6()
1046 {
1047 unsigned int result;
1048 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(6) :);
```

```
1049 return result;
1050 }
1051
1052 static __forceinline__ __device__ unsigned int optixGetPayload_7()
1053 {
1054 unsigned int result;
1055 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(7) :);
1056 return result;
1057 }
1058
1059 static __forceinline__ __device__ unsigned int optixGetPayload_8()
1060 {
1061 unsigned int result;
1062 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(8) :);
1063 return result;
1064 }
1065
1066 static __forceinline__ __device__ unsigned int optixGetPayload_9()
1067 {
1068 unsigned int result;
1069 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(9) :);
1070 return result;
1071 }
1072
1073 static __forceinline__ __device__ unsigned int optixGetPayload_10()
1074 {
1075 unsigned int result;
1076 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(10) :);
1077 return result;
1078 }
1079
1080 static __forceinline__ __device__ unsigned int optixGetPayload_11()
1081 {
1082 unsigned int result;
1083 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(11) :);
1084 return result;
1085 }
1086
1087 static __forceinline__ __device__ unsigned int optixGetPayload_12()
1088 {
1089 unsigned int result;
1090 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(12) :);
1091 return result;
1092 }
1093
1094 static __forceinline__ __device__ unsigned int optixGetPayload_13()
1095 {
1096 unsigned int result;
1097 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(13) :);
1098 return result;
1099 }
1100
1101 static __forceinline__ __device__ unsigned int optixGetPayload_14()
1102 {
1103 unsigned int result;
1104 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(14) :);
1105 return result;
1106 }
1107
1108 static __forceinline__ __device__ unsigned int optixGetPayload_15()
1109 {
1110 unsigned int result;
1111 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(15) :);
1112 return result;
1113 }
1114
1115 static __forceinline__ __device__ unsigned int optixGetPayload_16()
```

```
1116 {
1117 unsigned int result;
1118 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(16) :);
1119 return result;
1120 }
1121
1122 static __forceinline__ __device__ unsigned int optixGetPayload_17()
1123 {
1124 unsigned int result;
1125 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(17) :);
1126 return result;
1127 }
1128
1129 static __forceinline__ __device__ unsigned int optixGetPayload_18()
1130 {
1131 unsigned int result;
1132 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(18) :);
1133 return result;
1134 }
1135
1136 static __forceinline__ __device__ unsigned int optixGetPayload_19()
1137 {
1138 unsigned int result;
1139 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(19) :);
1140 return result;
1141 }
1142
1143 static __forceinline__ __device__ unsigned int optixGetPayload_20()
1144 {
1145 unsigned int result;
1146 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(20) :);
1147 return result;
1148 }
1149
1150 static __forceinline__ __device__ unsigned int optixGetPayload_21()
1151 {
1152 unsigned int result;
1153 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(21) :);
1154 return result;
1155 }
1156
1157 static __forceinline__ __device__ unsigned int optixGetPayload_22()
1158 {
1159 unsigned int result;
1160 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(22) :);
1161 return result;
1162 }
1163
1164 static __forceinline__ __device__ unsigned int optixGetPayload_23()
1165 {
1166 unsigned int result;
1167 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(23) :);
1168 return result;
1169 }
1170
1171 static __forceinline__ __device__ unsigned int optixGetPayload_24()
1172 {
1173 unsigned int result;
1174 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(24) :);
1175 return result;
1176 }
1177
1178 static __forceinline__ __device__ unsigned int optixGetPayload_25()
1179 {
1180 unsigned int result;
1181 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(25) :);
1182 return result;

```

```

1183 }
1184
1185 static __forceinline__ __device__ unsigned int optixGetPayload_26()
1186 {
1187 unsigned int result;
1188 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(26) :);
1189 return result;
1190 }
1191
1192 static __forceinline__ __device__ unsigned int optixGetPayload_27()
1193 {
1194 unsigned int result;
1195 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(27) :);
1196 return result;
1197 }
1198
1199 static __forceinline__ __device__ unsigned int optixGetPayload_28()
1200 {
1201 unsigned int result;
1202 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(28) :);
1203 return result;
1204 }
1205
1206 static __forceinline__ __device__ unsigned int optixGetPayload_29()
1207 {
1208 unsigned int result;
1209 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(29) :);
1210 return result;
1211 }
1212
1213 static __forceinline__ __device__ unsigned int optixGetPayload_30()
1214 {
1215 unsigned int result;
1216 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(30) :);
1217 return result;
1218 }
1219
1220 static __forceinline__ __device__ unsigned int optixGetPayload_31()
1221 {
1222 unsigned int result;
1223 asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(31) :);
1224 return result;
1225 }
1226
1227 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int types)
1228 {
1229 asm volatile("call _optix_set_payload_types, (%0);" : : "r"(types) :);
1230 }
1231
1232 static __forceinline__ __device__ unsigned int optixUndefinedValue()
1233 {
1234 unsigned int u0;
1235 asm("call (%0), _optix_undef_value, ();" : "=r"(u0) :);
1236 return u0;
1237 }
1238
1239 static __forceinline__ __device__ float3 optixGetWorldRayOrigin()
1240 {
1241 float f0, f1, f2;
1242 asm("call (%0), _optix_get_world_ray_origin_x, ();" : "=f"(f0) :);
1243 asm("call (%0), _optix_get_world_ray_origin_y, ();" : "=f"(f1) :);
1244 asm("call (%0), _optix_get_world_ray_origin_z, ();" : "=f"(f2) :);
1245 return make_float3(f0, f1, f2);
1246 }
1247
1248 static __forceinline__ __device__ float3 optixGetWorldRayDirection()
1249 {

```

```

1250 float f0, f1, f2;
1251 asm("call (%0), _optix_get_world_ray_direction_x, ();" : "=f"(f0) :);
1252 asm("call (%0), _optix_get_world_ray_direction_y, ();" : "=f"(f1) :);
1253 asm("call (%0), _optix_get_world_ray_direction_z, ();" : "=f"(f2) :);
1254 return make_float3(f0, f1, f2);
1255 }
1256
1257 static __forceinline__ __device__ float3 optixGetObjectRayOrigin()
1258 {
1259 float f0, f1, f2;
1260 asm("call (%0), _optix_get_object_ray_origin_x, ();" : "=f"(f0) :);
1261 asm("call (%0), _optix_get_object_ray_origin_y, ();" : "=f"(f1) :);
1262 asm("call (%0), _optix_get_object_ray_origin_z, ();" : "=f"(f2) :);
1263 return make_float3(f0, f1, f2);
1264 }
1265
1266 static __forceinline__ __device__ float3 optixGetObjectRayDirection()
1267 {
1268 float f0, f1, f2;
1269 asm("call (%0), _optix_get_object_ray_direction_x, ();" : "=f"(f0) :);
1270 asm("call (%0), _optix_get_object_ray_direction_y, ();" : "=f"(f1) :);
1271 asm("call (%0), _optix_get_object_ray_direction_z, ();" : "=f"(f2) :);
1272 return make_float3(f0, f1, f2);
1273 }
1274
1275 static __forceinline__ __device__ float optixGetRayTmin()
1276 {
1277 float f0;
1278 asm("call (%0), _optix_get_ray_tmin, ();" : "=f"(f0) :);
1279 return f0;
1280 }
1281
1282 static __forceinline__ __device__ float optixGetRayTmax()
1283 {
1284 float f0;
1285 asm("call (%0), _optix_get_ray_tmax, ();" : "=f"(f0) :);
1286 return f0;
1287 }
1288
1289 static __forceinline__ __device__ float optixGetRayTime()
1290 {
1291 float f0;
1292 asm("call (%0), _optix_get_ray_time, ();" : "=f"(f0) :);
1293 return f0;
1294 }
1295
1296 static __forceinline__ __device__ unsigned int optixGetRayFlags()
1297 {
1298 unsigned int u0;
1299 asm("call (%0), _optix_get_ray_flags, ();" : "=r"(u0) :);
1300 return u0;
1301 }
1302
1303 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask()
1304 {
1305 unsigned int u0;
1306 asm("call (%0), _optix_get_ray_visibility_mask, ();" : "=r"(u0) :);
1307 return u0;
1308 }
1309
1310 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias,
1311 unsigned int
instIdx)
1312 {
1313 unsigned long long handle;
1314 asm("call (%0), _optix_get_instance_traversable_from_ias, (%1, %2);"

```

```

1315 : "=l"(handle) : "l"(ias), "r"(instIdx));
1316 return (OptixTraversableHandle)handle;
1317 }
1318
1319
1320 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas,
1321 unsigned int primIdx,
1322 unsigned int sbtGASIndex,
1323 float time,
1324 float3 data[3])
1325 {
1326 asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_triangle_vertex_data, "
1327 "(%9, %10, %11, %12);"
1328 : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
1329 "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1330 : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1331 :);
1332 }
1333
1334 static __forceinline__ __device__ void optixGetMicroTriangleVertexData(float3 data[3])
1335 {
1336 asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_microtriangle_vertex_data, "
1337 "();"
1338 : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
1339 "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1340 :);
1341 }
1342 static __forceinline__ __device__ void optixGetMicroTriangleBarycentricsData(float2 data[3])
1343 {
1344 asm("call (%0, %1, %2, %3, %4, %5), _optix_get_microtriangle_barycentrics_data, "
1345 "();"
1346 : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[1].x), "=f"(data[1].y), "=f"(data[2].x),
1347 "=f"(data[2].y)
1348 :);
1349 }
1350 static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
1351 unsigned int primIdx,
1352 unsigned int sbtGASIndex,
1353 float time,
1354 float4 data[2])
1355 {
1356 asm("call (%0, %1, %2, %3, %4, %5, %6, %7), _optix_get_linear_curve_vertex_data, "
1357 "(%8, %9, %10, %11);"
1358 : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1359 "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
1360 : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1361 :);
1362 }
1363
1364 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(OptixTraversableHandle gas,
1365 unsigned int primIdx,
1366 unsigned int sbtGASIndex,
1367 float time,
1368 float4 data[3])
1369 {
1370 asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),
1371 _optix_get_quadratic_bspline_vertex_data, "
1372 "(%12, %13, %14, %15);"
1373 : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1374 "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1375 "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w)
1376 : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1377 :);
1378 }
1379 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,

```

```

1380 unsigned int primIdx,
1381 unsigned int sbtGASIndex,
1382 float time,
1383 float4 data[4])
1384 {
1385 asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1386 "_optix_get_cubic_bspline_vertex_data, "
1387 "(%16, %17, %18, %19);"
1388 : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1389 "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1390 "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
1391 "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
1392 : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1393 :);
1394 }
1395
1396 static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas,
1397 unsigned int primIdx,
1398 unsigned int sbtGASIndex,
1399 float time,
1400 float4 data[4])
1401 {
1402 asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1403 "_optix_get_catmullrom_vertex_data, "
1404 "(%16, %17, %18, %19);"
1405 : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1406 "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1407 "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1408 "=f"(data[3].w)
1409 : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1410 :);
1411 }
1412 static __forceinline__ __device__ void optixGetCubicBezierVertexData(OptixTraversableHandle gas,
1413 unsigned int primIdx,
1414 unsigned int sbtGASIndex,
1415 float time,
1416 float4 data[4])
1417 {
1418 asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1419 "_optix_get_cubic_bezier_vertex_data, "
1420 "(%16, %17, %18, %19);"
1421 : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1422 "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1423 "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1424 "=f"(data[3].w)
1425 : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1426 :);
1427 }
1428 static __forceinline__ __device__ void optixGetRibbonVertexData(OptixTraversableHandle gas,
1429 unsigned int primIdx,
1430 unsigned int sbtGASIndex,
1431 float time,
1432 float4 data[3])
1433 {
1434 asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11), _optix_get_ribbon_vertex_data, "
1435 "(%12, %13, %14, %15);"
1436 : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1437 "=f"(data[1].y),
1438 "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
1439 "=f"(data[2].w)
1440 : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1441 :);
1442 }
1443 static __forceinline__ __device__ float3 optixGetRibbonNormal(OptixTraversableHandle gas,

```

```

1443 unsigned int primIdx,
1444 unsigned int sbtGASIndex,
1445 float time,
1446 float2 ribbonParameters)
1447 {
1448 float3 normal;
1449 asm("call (%0, %1, %2), _optix_get_ribbon_normal, "
1450 "(%3, %4, %5, %6, %7, %8);"
1451 : "=f"(normal.x), "=f"(normal.y), "=f"(normal.z)
1452 : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time),
1453 "f"(ribbonParameters.x), "f"(ribbonParameters.y)
1454 :);
1455 return normal;
1456 }
1457
1458 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas,
1459 unsigned int primIdx,
1460 unsigned int sbtGASIndex,
1461 float time,
1462 float4 data[1])
1463 {
1464 asm("call (%0, %1, %2, %3), "
1465 "_optix_get_sphere_data, "
1466 "(%4, %5, %6, %7);"
1467 : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
1468 : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1469 :);
1470 }
1471
1472 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle()
1473 {
1474 unsigned long long handle;
1475 asm("call (%0), _optix_get_gas_traversable_handle, ();" : "=l"(handle) :);
1476 return (OptixTraversableHandle)handle;
1477 }
1478
1479 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle handle)
1480 {
1481 float f0;
1482 asm("call (%0), _optix_get_gas_motion_time_begin, (%1);" : "=f"(f0) : "l"(handle) :);
1483 return f0;
1484 }
1485
1486 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle handle)
1487 {
1488 float f0;
1489 asm("call (%0), _optix_get_gas_motion_time_end, (%1);" : "=f"(f0) : "l"(handle) :);
1490 return f0;
1491 }
1492
1493 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle handle)
1494 {
1495 unsigned int u0;
1496 asm("call (%0), _optix_get_gas_motion_step_count, (%1);" : "=r"(u0) : "l"(handle) :);
1497 return u0;
1498 }
1499
1500 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12])
1501 {
1502 if(optixGetTransformListSize() == 0)
1503 {
1504 m[0] = 1.0f;
1505 m[1] = 0.0f;
1506 m[2] = 0.0f;
1507 m[3] = 0.0f;
1508 m[4] = 0.0f;
1509 m[5] = 1.0f;

```

```
1510 m[6] = 0.0f;
1511 m[7] = 0.0f;
1512 m[8] = 0.0f;
1513 m[9] = 0.0f;
1514 m[10] = 1.0f;
1515 m[11] = 0.0f;
1516 return;
1517 }
1518
1519 float4 m0, m1, m2;
1520 optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
1521 m[0] = m0.x;
1522 m[1] = m0.y;
1523 m[2] = m0.z;
1524 m[3] = m0.w;
1525 m[4] = m1.x;
1526 m[5] = m1.y;
1527 m[6] = m1.z;
1528 m[7] = m1.w;
1529 m[8] = m2.x;
1530 m[9] = m2.y;
1531 m[10] = m2.z;
1532 m[11] = m2.w;
1533 }
1534
1535 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12])
1536 {
1537 if(optixGetTransformListSize() == 0)
1538 {
1539 m[0] = 1.0f;
1540 m[1] = 0.0f;
1541 m[2] = 0.0f;
1542 m[3] = 0.0f;
1543 m[4] = 0.0f;
1544 m[5] = 1.0f;
1545 m[6] = 0.0f;
1546 m[7] = 0.0f;
1547 m[8] = 0.0f;
1548 m[9] = 0.0f;
1549 m[10] = 1.0f;
1550 m[11] = 0.0f;
1551 return;
1552 }
1553
1554 float4 m0, m1, m2;
1555 optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);
1556 m[0] = m0.x;
1557 m[1] = m0.y;
1558 m[2] = m0.z;
1559 m[3] = m0.w;
1560 m[4] = m1.x;
1561 m[5] = m1.y;
1562 m[6] = m1.z;
1563 m[7] = m1.w;
1564 m[8] = m2.x;
1565 m[9] = m2.y;
1566 m[10] = m2.z;
1567 m[11] = m2.w;
1568 }
1569
1570 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point)
1571 {
1572 if(optixGetTransformListSize() == 0)
1573 return point;
1574
1575 float4 m0, m1, m2;
1576 optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
```

```

1577 return optix_impl::optixTransformPoint(m0, m1, m2, point);
1578 }
1579
1580 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec)
1581 {
1582 if(optixGetTransformListSize() == 0)
1583 return vec;
1584
1585 float4 m0, m1, m2;
1586 optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
1587 return optix_impl::optixTransformVector(m0, m1, m2, vec);
1588 }
1589
1590 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal)
1591 {
1592 if(optixGetTransformListSize() == 0)
1593 return normal;
1594
1595 float4 m0, m1, m2;
1596 optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2); // inverse of
optixGetWorldToObjectTransformMatrix()
1597 return optix_impl::optixTransformNormal(m0, m1, m2, normal);
1598 }
1599
1600 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point)
1601 {
1602 if(optixGetTransformListSize() == 0)
1603 return point;
1604
1605 float4 m0, m1, m2;
1606 optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);
1607 return optix_impl::optixTransformPoint(m0, m1, m2, point);
1608 }
1609
1610 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec)
1611 {
1612 if(optixGetTransformListSize() == 0)
1613 return vec;
1614
1615 float4 m0, m1, m2;
1616 optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);
1617 return optix_impl::optixTransformVector(m0, m1, m2, vec);
1618 }
1619
1620 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal)
1621 {
1622 if(optixGetTransformListSize() == 0)
1623 return normal;
1624
1625 float4 m0, m1, m2;
1626 optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2); // inverse of
optixGetObjectToWorldTransformMatrix()
1627 return optix_impl::optixTransformNormal(m0, m1, m2, normal);
1628 }
1629
1630 static __forceinline__ __device__ unsigned int optixGetTransformListSize()
1631 {
1632 unsigned int u0;
1633 asm("call (%0), _optix_get_transform_list_size, ();" : "=r"(u0) :);
1634 return u0;
1635 }
1636
1637 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int index)
1638 {
1639 unsigned long long u0;
1640 asm("call (%0), _optix_get_transform_list_handle, (%1);" : "=l"(u0) : "r"(index) :);
1641 return u0;

```

```

1642 }
1643
1644 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle)
1645 {
1646 int i0;
1647 asm("call (%0), _optix_get_transform_type_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
1648 return (OptixTransformType)i0;
1649 }
1650
1651 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle)
1652 {
1653 unsigned long long ptr;
1654 asm("call (%0), _optix_get_static_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
1655 return (const OptixStaticTransform*)ptr;
1656 }
1657
1658 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle)
1659 {
1660 unsigned long long ptr;
1661 asm("call (%0), _optix_get_srt_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
1662 return (const OptixSRTMotionTransform*)ptr;
1663 }
1664
1665 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle)
1666 {
1667 unsigned long long ptr;
1668 asm("call (%0), _optix_get_matrix_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
1669 return (const OptixMatrixMotionTransform*)ptr;
1670 }
1671
1672 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle)
1673 {
1674 int i0;
1675 asm("call (%0), _optix_get_instance_id_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
1676 return i0;
1677 }
1678
1679 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle(OptixTraversableHandle handle)
1680 {
1681 unsigned long long i0;
1682 asm("call (%0), _optix_get_instance_child_from_handle, (%1);" : "=l"(i0) : "l"(handle) :);
1683 return (OptixTraversableHandle)i0;
1684 }
1685
1686 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle)
1687 {
1688 unsigned long long ptr;
1689 asm("call (%0), _optix_get_instance_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
1690 return (const float4*)ptr;
1691 }
1692
1693 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle)
1694 {
1695 unsigned long long ptr;
1696 asm("call (%0), _optix_get_instance_inverse_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle)
:);
1697 return (const float4*)ptr;
1698 }
1699

```

```

1700 static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle(OptixTraversableHandle
handle)
1701 {
1702 unsigned long long ptr;
1703 asm("call (%0), _optix_get_gas_ptr_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
1704 return (CUdeviceptr)ptr;
1705 }
1706 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind)
1707 {
1708 int ret;
1709 asm volatile(
1710 "call (%0), _optix_report_intersection_0"
1711 ", (%1, %2);"
1712 : "=r"(ret)
1713 : "f"(hitT), "r"(hitKind)
1714 :);
1715 return ret;
1716 }
1717
1718 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0)
1719 {
1720 int ret;
1721 asm volatile(
1722 "call (%0), _optix_report_intersection_1"
1723 ", (%1, %2, %3);"
1724 : "=r"(ret)
1725 : "f"(hitT), "r"(hitKind), "r"(a0)
1726 :);
1727 return ret;
1728 }
1729
1730 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1)
1731 {
1732 int ret;
1733 asm volatile(
1734 "call (%0), _optix_report_intersection_2"
1735 ", (%1, %2, %3, %4);"
1736 : "=r"(ret)
1737 : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1)
1738 :);
1739 return ret;
1740 }
1741
1742 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1, unsigned int a2)
1743 {
1744 int ret;
1745 asm volatile(
1746 "call (%0), _optix_report_intersection_3"
1747 ", (%1, %2, %3, %4, %5);"
1748 : "=r"(ret)
1749 : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2)
1750 :);
1751 return ret;
1752 }
1753
1754 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
1755 unsigned int hitKind,
1756 unsigned int a0,
1757 unsigned int a1,
1758 unsigned int a2,
1759 unsigned int a3)
1760 {
1761 int ret;
1762 asm volatile(

```

```

1763 "call (%0), _optix_report_intersection_4"
1764 ", (%1, %2, %3, %4, %5, %6);"
1765 : "=r"(ret)
1766 : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3)
1767 :);
1768 return ret;
1769 }
1770
1771 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
1772 unsigned int hitKind,
1773 unsigned int a0,
1774 unsigned int a1,
1775 unsigned int a2,
1776 unsigned int a3,
1777 unsigned int a4)
1778 {
1779 int ret;
1780 asm volatile(
1781 "call (%0), _optix_report_intersection_5"
1782 ", (%1, %2, %3, %4, %5, %6, %7);"
1783 : "=r"(ret)
1784 : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4)
1785 :);
1786 return ret;
1787 }
1788
1789 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
1790 unsigned int hitKind,
1791 unsigned int a0,
1792 unsigned int a1,
1793 unsigned int a2,
1794 unsigned int a3,
1795 unsigned int a4,
1796 unsigned int a5)
1797 {
1798 int ret;
1799 asm volatile(
1800 "call (%0), _optix_report_intersection_6"
1801 ", (%1, %2, %3, %4, %5, %6, %7, %8);"
1802 : "=r"(ret)
1803 : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5)
1804 :);
1805 return ret;
1806 }
1807
1808 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
1809 unsigned int hitKind,
1810 unsigned int a0,
1811 unsigned int a1,
1812 unsigned int a2,
1813 unsigned int a3,
1814 unsigned int a4,
1815 unsigned int a5,
1816 unsigned int a6)
1817 {
1818 int ret;
1819 asm volatile(
1820 "call (%0), _optix_report_intersection_7"
1821 ", (%1, %2, %3, %4, %5, %6, %7, %8, %9);"
1822 : "=r"(ret)
1823 : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6)
1824 :);
1825 return ret;
1826 }
1827
1828 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
1829 unsigned int hitKind,

```

```

1830 unsigned int a0,
1831 unsigned int a1,
1832 unsigned int a2,
1833 unsigned int a3,
1834 unsigned int a4,
1835 unsigned int a5,
1836 unsigned int a6,
1837 unsigned int a7)
1838 {
1839 int ret;
1840 asm volatile(
1841 "call (%0), _optix_report_intersection_8"
1842 ", (%1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
1843 : "=r"(ret)
1844 : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6), "r"(a7)
1845 :);
1846 return ret;
1847 }
1848
1849 #define OPTIX_DEFINE_optixGetAttribute_BODY(which)
1850 \
1851 unsigned int ret;
1852 \
1853 asm("call (%0), _optix_get_attribute_" #which ", ();" : "=r"(ret) :);
1854 \
1855 return ret;
1856
1857 static __forceinline__ __device__ unsigned int optixGetAttribute_0()
1858 {
1859 OPTIX_DEFINE_optixGetAttribute_BODY(0);
1860 }
1861
1862 static __forceinline__ __device__ unsigned int optixGetAttribute_1()
1863 {
1864 OPTIX_DEFINE_optixGetAttribute_BODY(1);
1865 }
1866
1867 static __forceinline__ __device__ unsigned int optixGetAttribute_2()
1868 {
1869 OPTIX_DEFINE_optixGetAttribute_BODY(2);
1870 }
1871
1872 static __forceinline__ __device__ unsigned int optixGetAttribute_3()
1873 {
1874 OPTIX_DEFINE_optixGetAttribute_BODY(3);
1875 }
1876
1877 static __forceinline__ __device__ unsigned int optixGetAttribute_4()
1878 {
1879 OPTIX_DEFINE_optixGetAttribute_BODY(4);
1880 }
1881
1882 static __forceinline__ __device__ unsigned int optixGetAttribute_5()
1883 {
1884 OPTIX_DEFINE_optixGetAttribute_BODY(5);
1885 }
1886
1887 static __forceinline__ __device__ unsigned int optixGetAttribute_6()
1888 {
1889 OPTIX_DEFINE_optixGetAttribute_BODY(6);
1890 }
1891
1892 static __forceinline__ __device__ unsigned int optixGetAttribute_7()
1893 {
1894 OPTIX_DEFINE_optixGetAttribute_BODY(7);
1895 }

```

```

1894 #undef OPTIX_DEFINE_optixGetAttribute_BODY
1895
1896 static __forceinline__ __device__ void optixTerminateRay()
1897 {
1898 asm volatile("call _optix_terminate_ray, ();");
1899 }
1900
1901 static __forceinline__ __device__ void optixIgnoreIntersection()
1902 {
1903 asm volatile("call _optix_ignore_intersection, ();");
1904 }
1905
1906 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex()
1907 {
1908 unsigned int u0;
1909 asm("call (%0), _optix_read_primitive_idx, ();" : "=r"(u0) :);
1910 return u0;
1911 }
1912
1913 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex()
1914 {
1915 unsigned int u0;
1916 asm("call (%0), _optix_read_sbt_gas_idx, ();" : "=r"(u0) :);
1917 return u0;
1918 }
1919
1920 static __forceinline__ __device__ unsigned int optixGetInstanceId()
1921 {
1922 unsigned int u0;
1923 asm("call (%0), _optix_read_instance_id, ();" : "=r"(u0) :);
1924 return u0;
1925 }
1926
1927 static __forceinline__ __device__ unsigned int optixGetInstanceIndex()
1928 {
1929 unsigned int u0;
1930 asm("call (%0), _optix_read_instance_idx, ();" : "=r"(u0) :);
1931 return u0;
1932 }
1933
1934 static __forceinline__ __device__ unsigned int optixGetHitKind()
1935 {
1936 unsigned int u0;
1937 asm("call (%0), _optix_get_hit_kind, ();" : "=r"(u0) :);
1938 return u0;
1939 }
1940
1941 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind)
1942 {
1943 unsigned int u0;
1944 asm("call (%0), _optix_get_primitive_type_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));
1945 return (OptixPrimitiveType)u0;
1946 }
1947
1948 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind)
1949 {
1950 unsigned int u0;
1951 asm("call (%0), _optix_get_backface_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));
1952 return (u0 == 0x1);
1953 }
1954
1955 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind)
1956 {
1957 return !optixIsBackFaceHit(hitKind);
1958 }
1959
1960

```

```

1961 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType()
1962 {
1963 return optixGetPrimitiveType(optixGetHitKind());
1964 }
1965
1966 static __forceinline__ __device__ bool optixIsBackFaceHit()
1967 {
1968 return optixIsBackFaceHit(optixGetHitKind());
1969 }
1970
1971 static __forceinline__ __device__ bool optixIsFrontFaceHit()
1972 {
1973 return optixIsFrontFaceHit(optixGetHitKind());
1974 }
1975
1976 static __forceinline__ __device__ bool optixIsTriangleHit()
1977 {
1978 return optixIsTriangleFrontFaceHit() || optixIsTriangleBackFaceHit();
1979 }
1980
1981 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit()
1982 {
1983 return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE;
1984 }
1985
1986 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit()
1987 {
1988 return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_BACK_FACE;
1989 }
1990
1991 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit()
1992 {
1993 return optixGetPrimitiveType(optixGetHitKind()) ==
OPTIX_PRIMITIVE_TYPE_DISPLACED_MICROMESH_TRIANGLE;
1994 }
1995
1996 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleFrontFaceHit()
1997 {
1998 return optixIsDisplacedMicromeshTriangleHit() && optixIsFrontFaceHit();
1999 }
2000
2001 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleBackFaceHit()
2002 {
2003 return optixIsDisplacedMicromeshTriangleHit() && optixIsBackFaceHit();
2004 }
2005
2006 static __forceinline__ __device__ float optixGetCurveParameter()
2007 {
2008 float f0;
2009 asm("call (%0), _optix_get_curve_parameter, ();" : "=f"(f0) :);
2010 return f0;
2011 }
2012
2013 static __forceinline__ __device__ float2 optixGetRibbonParameters()
2014 {
2015 float f0, f1;
2016 asm("call (%0, %1), _optix_get_ribbon_parameters, ();" : "=f"(f0), "=f"(f1) :);
2017 return make_float2(f0, f1);
2018 }
2019
2020 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics()
2021 {
2022 float f0, f1;
2023 asm("call (%0, %1), _optix_get_triangle_barycentrics, ();" : "=f"(f0), "=f"(f1) :);
2024 return make_float2(f0, f1);
2025 }
2026

```

```

2027 static __forceinline__ __device__ uint3 optixGetLaunchIndex()
2028 {
2029 unsigned int u0, u1, u2;
2030 asm("call (%0), _optix_get_launch_index_x, ();" : "=r"(u0) :);
2031 asm("call (%0), _optix_get_launch_index_y, ();" : "=r"(u1) :);
2032 asm("call (%0), _optix_get_launch_index_z, ();" : "=r"(u2) :);
2033 return make_uint3(u0, u1, u2);
2034 }
2035
2036 static __forceinline__ __device__ uint3 optixGetLaunchDimensions()
2037 {
2038 unsigned int u0, u1, u2;
2039 asm("call (%0), _optix_get_launch_dimension_x, ();" : "=r"(u0) :);
2040 asm("call (%0), _optix_get_launch_dimension_y, ();" : "=r"(u1) :);
2041 asm("call (%0), _optix_get_launch_dimension_z, ();" : "=r"(u2) :);
2042 return make_uint3(u0, u1, u2);
2043 }
2044
2045 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer()
2046 {
2047 unsigned long long ptr;
2048 asm("call (%0), _optix_get_sbt_data_ptr_64, ();" : "=l"(ptr) :);
2049 return (CUdeviceptr)ptr;
2050 }
2051
2052 static __forceinline__ __device__ void optixThrowException(int exceptionCode)
2053 {
2054 asm volatile(
2055 "call _optix_throw_exception_0, (%0);"
2056 : /* no return value */
2057 : "r"(exceptionCode)
2058 :);
2059 }
2060
2061 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0)
2062 {
2063 asm volatile(
2064 "call _optix_throw_exception_1, (%0, %1);"
2065 : /* no return value */
2066 : "r"(exceptionCode), "r"(exceptionDetail0)
2067 :);
2068 }
2069
2070 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1)
2071 {
2072 asm volatile(
2073 "call _optix_throw_exception_2, (%0, %1, %2);"
2074 : /* no return value */
2075 : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1)
2076 :);
2077 }
2078
2079 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
2080 {
2081 asm volatile(
2082 "call _optix_throw_exception_3, (%0, %1, %2, %3);"
2083 : /* no return value */
2084 : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2)
2085 :);
2086 }
2087
2088 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3)

```

```

2089 {
2090 asm volatile(
2091 "call _optix_throw_exception_4, (%0, %1, %2, %3, %4);"
2092 : /* no return value */
2093 : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
2094 "r"(exceptionDetail3)
2095 :);
2096 }
2097 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3, unsigned int exceptionDetail4)
2098 {
2099 asm volatile(
2100 "call _optix_throw_exception_5, (%0, %1, %2, %3, %4, %5);"
2101 : /* no return value */
2102 : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
2103 "r"(exceptionDetail3), "r"(exceptionDetail4)
2104 :);
2105 }
2106 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)
2107 {
2108 asm volatile(
2109 "call _optix_throw_exception_6, (%0, %1, %2, %3, %4, %5, %6);"
2110 : /* no return value */
2111 : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
2112 "r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5)
2113 :);
2114 }
2115 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)
2116 {
2117 asm volatile(
2118 "call _optix_throw_exception_7, (%0, %1, %2, %3, %4, %5, %6, %7);"
2119 : /* no return value */
2120 : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
2121 "r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6)
2122 :);
2123 }
2124 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)
2125 {
2126 asm volatile(
2127 "call _optix_throw_exception_8, (%0, %1, %2, %3, %4, %5, %6, %7, %8);"
2128 : /* no return value */
2129 : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
2130 "r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6),
2131 "r"(exceptionDetail7)
2132 :);
2133 }
2133 static __forceinline__ __device__ int optixGetExceptionCode()
2134 {
2135 int s0;
2136 asm("call (%0), _optix_get_exception_code, ();" : "=r"(s0) :);
2137 return s0;
2138 }
2139

```

```

2140 #define OPTIX_DEFINE_optixGetExceptionDetail_BODY(which)
\
2141 unsigned int ret;
\
2142 asm("call (%0), _optix_get_exception_detail_" #which ", ();" : "=r"(ret) :);
\
2143 return ret;
2144
2145 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0()
2146 {
2147 OPTIX_DEFINE_optixGetExceptionDetail_BODY(0);
2148 }
2149
2150 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1()
2151 {
2152 OPTIX_DEFINE_optixGetExceptionDetail_BODY(1);
2153 }
2154
2155 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2()
2156 {
2157 OPTIX_DEFINE_optixGetExceptionDetail_BODY(2);
2158 }
2159
2160 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3()
2161 {
2162 OPTIX_DEFINE_optixGetExceptionDetail_BODY(3);
2163 }
2164
2165 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4()
2166 {
2167 OPTIX_DEFINE_optixGetExceptionDetail_BODY(4);
2168 }
2169
2170 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5()
2171 {
2172 OPTIX_DEFINE_optixGetExceptionDetail_BODY(5);
2173 }
2174
2175 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6()
2176 {
2177 OPTIX_DEFINE_optixGetExceptionDetail_BODY(6);
2178 }
2179
2180 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7()
2181 {
2182 OPTIX_DEFINE_optixGetExceptionDetail_BODY(7);
2183 }
2184
2185 #undef OPTIX_DEFINE_optixGetExceptionDetail_BODY
2186
2187
2188 static __forceinline__ __device__ char* optixGetExceptionLineInfo()
2189 {
2190 unsigned long long ptr;
2191 asm("call (%0), _optix_get_exception_line_info, ();" : "=l"(ptr) :);
2192 return (char*)ptr;
2193 }
2194
2195 template <typename ReturnT, typename... ArgTypes>
2196 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args)
2197 {
2198 unsigned long long func;
2199 asm("call (%0), _optix_call_direct_callable,(%1);" : "=l"(func) : "r"(sbtIndex) :);
2200 using funcT = ReturnT (*)(ArgTypes...);
2201 funcT call = (funcT)(func);
2202 return call(args...);
2203 }

```

```

2204
2205 template <typename ReturnT, typename... ArgTypes>
2206 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes... args)
2207 {
2208 unsigned long long func;
2209 asm("call (%0), _optix_call_continuation_callable,(%1);" : "=l"(func) : "r"(sbtIndex) :);
2210 using funcT = ReturnT (*)(ArgTypes...);
2211 funcT call = (funcT)(func);
2212 return call(args...);
2213 }
2214
2215 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int
texInfo, float x, float y, unsigned int* singleMipLevel)
2216 {
2217 uint4 result;
2218 unsigned long long resultPtr = reinterpret_cast<unsigned long long>(&result);
2219 unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2220 // Cast float args to integers, because the intrinsics take .b32 arguments when compiled to PTX.
2221 asm volatile(
2222 "call _optix_tex_footprint_2d_v2"
2223 ", (%0, %1, %2, %3, %4, %5);"
2224 :
2225 : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2226 "l"(singleMipLevelPtr), "l"(resultPtr)
2227 :);
2228 return result;
2229 }
2230
2231 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
2232 unsigned int texInfo,
2233 float x,
2234 float y,
2235 float dPdx_x,
2236 float dPdx_y,
2237 float dPdy_x,
2238 float dPdy_y,
2239 bool coarse,
2240 unsigned int* singleMipLevel)
2241 {
2242 uint4 result;
2243 unsigned long long resultPtr = reinterpret_cast<unsigned long long>(&result);
2244 unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2245 // Cast float args to integers, because the intrinsics take .b32 arguments when compiled to PTX.
2246 asm volatile(
2247 "call _optix_tex_footprint_2d_grad_v2"
2248 ", (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
2249 :
2250 : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2251 "r"(__float_as_uint(dPdx_x)), "r"(__float_as_uint(dPdx_y)), "r"(__float_as_uint(dPdy_x)),
2252 "r"(__float_as_uint(dPdy_y)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
2253 "l"(resultPtr)
2254 :);
2255 return result;
2256 }
2257
2258 static __forceinline__ __device__ uint4
2259 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool
coarse, unsigned int* singleMipLevel)
2260 {
2261 uint4 result;
2262 unsigned long long resultPtr = reinterpret_cast<unsigned long long>(&result);
2263 unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2264 // Cast float args to integers, because the intrinsics take .b32 arguments when compiled to PTX.
2265 asm volatile(
2266 "call _optix_tex_footprint_2d_lod_v2"
2267 ", (%0, %1, %2, %3, %4, %5, %6, %7);"

```

```

2268 :
2269 : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2270 "r"(__float_as_uint(level)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
"l"(resultPtr)
2271 :);
2272 return result;
2273 }
2274
2275 #endif // OPTIX_DEVICE_IMPL_H

```

## 8.3 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)

### 8.3.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Device side implementation for transformation helper functions.

## 8.4 optix\_device\_impl\_transformations.h

Go to the documentation of this file.

```

1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
4 *
5 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
6 * property and proprietary rights in and to this material, related
7 * documentation and any modifications thereto. Any use, reproduction,
8 * disclosure or distribution of this material and related documentation
9 * without an express license agreement from NVIDIA CORPORATION or
10 * its affiliates is strictly prohibited.
11 */
12 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
13 #error("optix_device_impl_transformations.h is an internal header file and must not be used directly.
14 Please use optix_device.h or optix.h instead.")
15 #endif
16
17 #ifndef OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H
18 #define OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H
19
20 namespace optix_impl {
21
22 static __forceinline__ __device__ float4 optixAddFloat4(const float4& a, const float4& b)
23 {
24 return make_float4(a.x + b.x, a.y + b.y, a.z + b.z, a.w + b.w);
25 }
26
27 static __forceinline__ __device__ float4 optixMulFloat4(const float4& a, float b)
28 {
29 return make_float4(a.x * b, a.y * b, a.z * b, a.w * b);
30 }
31
32 static __forceinline__ __device__ uint4 optixLdg(unsigned long long addr)
33 {
34 const uint4* ptr;
35 asm volatile("cvta.to.global.u64 %0, %1;" : "=l"(ptr) : "l"(addr));
36 uint4 ret;
37 asm volatile("ld.global.v4.u32 {%0,%1,%2,%3}, [%4];"
38 : "=r"(ret.x), "=r"(ret.y), "=r"(ret.z), "=r"(ret.w)
39 : "l"(ptr));
40 return ret;
41 }
42
43 template <class T>
44 static __forceinline__ __device__ T optixLoadReadOnlyAlign16(const T* ptr)
45 {
46 // Debug mode may keep this temporary variable
47 // If T does not enforce 16B alignment, v may not be 16B aligned and storing the loaded data from ptr
48 // fails
49 __align__(16) T v;
50 for(int ofs = 0; ofs < sizeof(T); ofs += 16)
51 *(uint4*)((char*)&v + ofs) = optixLdg((unsigned long long)((char*)ptr + ofs));
52 return v;
53 }

```

```

59 }
60
61 // Multiplies the row vector vec with the 3x4 matrix with rows m0, m1, and m2
62 static __forceinline__ __device__ float4 optixMultiplyRowMatrix(const float4 vec, const float4 m0, const
float4 m1, const float4 m2)
63 {
64 float4 result;
65
66 result.x = vec.x * m0.x + vec.y * m1.x + vec.z * m2.x;
67 result.y = vec.x * m0.y + vec.y * m1.y + vec.z * m2.y;
68 result.z = vec.x * m0.z + vec.y * m1.z + vec.z * m2.z;
69 result.w = vec.x * m0.w + vec.y * m1.w + vec.z * m2.w + vec.w;
70
71 return result;
72 }
73
74 // Converts the SRT transformation srt into a 3x4 matrix with rows m0, m1, and m2
75 static __forceinline__ __device__ void optixGetMatrixFromSrt(float4& m0, float4& m1, float4& m2, const
OptixSRTData& srt)
76 {
77 // assumed to be normalized
78 const float4 q = {srt.qx, srt.qy, srt.qz, srt.qw};
79
80 const float sqw = q.w * q.w;
81 const float sqx = q.x * q.x;
82 const float sqy = q.y * q.y;
83 const float sqz = q.z * q.z;
84
85 const float xy = q.x * q.y;
86 const float zw = q.z * q.w;
87 const float xz = q.x * q.z;
88 const float yw = q.y * q.w;
89 const float yz = q.y * q.z;
90 const float xw = q.x * q.w;
91
92 m0.x = (sqx - sqy - sqz + sqw);
93 m0.y = 2.0f * (xy - zw);
94 m0.z = 2.0f * (xz + yw);
95
96 m1.x = 2.0f * (xy + zw);
97 m1.y = (-sqx + sqy - sqz + sqw);
98 m1.z = 2.0f * (yz - xw);
99
100 m2.x = 2.0f * (xz - yw);
101 m2.y = 2.0f * (yz + xw);
102 m2.z = (-sqx - sqy + sqz + sqw);
103
104 m0.w = m0.x * srt.pvx + m0.y * srt.pvy + m0.z * srt.pvz + srt.tx;
105 m1.w = m1.x * srt.pvx + m1.y * srt.pvy + m1.z * srt.pvz + srt.ty;
106 m2.w = m2.x * srt.pvx + m2.y * srt.pvy + m2.z * srt.pvz + srt.tz;
107
108 m0.z = m0.x * srt.b + m0.y * srt.c + m0.z * srt.sz;
109 m1.z = m1.x * srt.b + m1.y * srt.c + m1.z * srt.sz;
110 m2.z = m2.x * srt.b + m2.y * srt.c + m2.z * srt.sz;
111
112 m0.y = m0.x * srt.a + m0.y * srt.sy;
113 m1.y = m1.x * srt.a + m1.y * srt.sy;
114 m2.y = m2.x * srt.a + m2.y * srt.sy;
115
116 m0.x = m0.x * srt.sx;
117 m1.x = m1.x * srt.sx;
118 m2.x = m2.x * srt.sx;
119 }
120
121 // Inverts a 3x4 matrix in place
122 static __forceinline__ __device__ void optixInvertMatrix(float4& m0, float4& m1, float4& m2)
123 {

```

```

124 const float det3 =
125 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 -
m1.y * m2.x);
126
127 const float inv_det3 = 1.0f / det3;
128
129 float inv3[3][3];
130 inv3[0][0] = inv_det3 * (m1.y * m2.z - m2.y * m1.z);
131 inv3[0][1] = inv_det3 * (m0.z * m2.y - m2.z * m0.y);
132 inv3[0][2] = inv_det3 * (m0.y * m1.z - m1.y * m0.z);
133
134 inv3[1][0] = inv_det3 * (m1.z * m2.x - m2.z * m1.x);
135 inv3[1][1] = inv_det3 * (m0.x * m2.z - m2.x * m0.z);
136 inv3[1][2] = inv_det3 * (m0.z * m1.x - m1.z * m0.x);
137
138 inv3[2][0] = inv_det3 * (m1.x * m2.y - m2.x * m1.y);
139 inv3[2][1] = inv_det3 * (m0.y * m2.x - m2.y * m0.x);
140 inv3[2][2] = inv_det3 * (m0.x * m1.y - m1.x * m0.y);
141
142 const float b[3] = {m0.w, m1.w, m2.w};
143
144 m0.x = inv3[0][0];
145 m0.y = inv3[0][1];
146 m0.z = inv3[0][2];
147 m0.w = -inv3[0][0] * b[0] - inv3[0][1] * b[1] - inv3[0][2] * b[2];
148
149 m1.x = inv3[1][0];
150 m1.y = inv3[1][1];
151 m1.z = inv3[1][2];
152 m1.w = -inv3[1][0] * b[0] - inv3[1][1] * b[1] - inv3[1][2] * b[2];
153
154 m2.x = inv3[2][0];
155 m2.y = inv3[2][1];
156 m2.z = inv3[2][2];
157 m2.w = -inv3[2][0] * b[0] - inv3[2][1] * b[1] - inv3[2][2] * b[2];
158 }
159
160 static __forceinline__ __device__ void optixLoadInterpolatedMatrixKey(float4& m0, float4& m1, float4&
m2, const float4* matrix, const float t1)
161 {
162 m0 = optixLoadReadOnlyAlign16(&matrix[0]);
163 m1 = optixLoadReadOnlyAlign16(&matrix[1]);
164 m2 = optixLoadReadOnlyAlign16(&matrix[2]);
165
166 // The conditional prevents concurrent loads leading to spills
167 if(t1 > 0.0f)
168 {
169 const float t0 = 1.0f - t1;
170 m0 = optixAddFloat4(optixMulFloat4(m0, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[3]),
t1));
171 m1 = optixAddFloat4(optixMulFloat4(m1, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[4]),
t1));
172 m2 = optixAddFloat4(optixMulFloat4(m2, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[5]),
t1));
173 }
174 }
175
176 static __forceinline__ __device__ void optixLoadInterpolatedSrtKey(float4& srt0,
177 float4& srt1,
178 float4& srt2,
179 float4& srt3,
180 const float4* srt,
181 const float t1)
182 {
183 srt0 = optixLoadReadOnlyAlign16(&srt[0]);
184 srt1 = optixLoadReadOnlyAlign16(&srt[1]);
185 srt2 = optixLoadReadOnlyAlign16(&srt[2]);

```

```

186 srt3 = optixLoadReadOnlyAlign16(&srt[3]);
187
188 // The conditional prevents concurrent loads leading to spills
189 if(t1 > 0.0f)
190 {
191 const float t0 = 1.0f - t1;
192 srt0 = optixAddFloat4(optixMulFloat4(srt0, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[4]),
193 t1));
194 srt1 = optixAddFloat4(optixMulFloat4(srt1, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[5]),
195 t1));
196 srt2 = optixAddFloat4(optixMulFloat4(srt2, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[6]),
197 t1));
198 srt3 = optixAddFloat4(optixMulFloat4(srt3, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[7]),
199 t1));
200
201 float inv_length = 1.f / sqrt(srt2.y * srt2.y + srt2.z * srt2.z + srt2.w * srt2.w + srt3.x *
202 srt3.x);
203 srt2.y *= inv_length;
204 srt2.z *= inv_length;
205 srt2.w *= inv_length;
206 srt3.x *= inv_length;
207 }
208 }
209
210 static __forceinline__ __device__ void optixResolveMotionKey(float& localt, int& key, const
211 OptixMotionOptions& options, const float globalt)
212 {
213 const float timeBegin = options.timeBegin;
214 const float timeEnd = options.timeEnd;
215 const float numIntervals = (float)(options.numKeys - 1);
216
217 // No need to check the motion flags. If data originates from a valid transform list handle, then
218 globalt is in
219 // range, or vanish flags are not set.
220
221 // should be NaN or in [0,numIntervals]
222 float time = max(0.f, min(numIntervals, numIntervals * __fdivdef(globalt - timeBegin, timeEnd -
223 timeBegin)));
224
225 // catch NaN (for example when timeBegin=timeEnd)
226 if(time != time)
227 time = 0.f;
228
229 const float fltKey = fminf(floorf(time), numIntervals - 1);
230
231 localt = time - fltKey;
232 key = (int)fltKey;
233 }
234
235 // Returns the interpolated transformation matrix for a particular matrix motion transformation and point
236 in time.
237 static __forceinline__ __device__ void optixGetInterpolatedTransformation(float4&
238 trf0,
239 float4& trf1,
240 float4& trf2,
241 const OptixMatrixMotionTransform*
242 transformData,
243 const float time)
244 {
245 // Compute key and intra key time
246 float keyTime;
247 int key;
248 optixResolveMotionKey(keyTime, key, optixLoadReadOnlyAlign16(transformData).motionOptions, time);
249
250 // Get pointer to left key
251 const float4* transform = (const float4*)&transformData->transform[key][0];
252 }

```

```

242 // Load and interpolate matrix keys
243 optixLoadInterpolatedMatrixKey(trf0, trf1, trf2, transform, keyTime);
244 }
245
246 // Returns the interpolated transformation matrix for a particular SRT motion transformation and point in
time.
247 static __forceinline__ __device__ void optixGetInterpolatedTransformation(float4&
trf0,
248 float4& trf1,
249 float4& trf2,
250 const OptixSRTMotionTransform*
transformData,
251 const float time)
252 {
253 // Compute key and intra key time
254 float keyTime;
255 int key;
256 optixResolveMotionKey(keyTime, key, optixLoadReadOnlyAlign16(transformData).motionOptions, time);
257
258 // Get pointer to left key
259 const float4* dataPtr = reinterpret_cast<const float4*>(&transformData->srtData[key]);
260
261 // Load and interpolated SRT keys
262 float4 data[4];
263 optixLoadInterpolatedSrtKey(data[0], data[1], data[2], data[3], dataPtr, keyTime);
264
265 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,
266 data[2].x, data[2].y, data[2].z, data[2].w, data[3].x, data[3].y, data[3].z,
data[3].w};
267
268 // Convert SRT into a matrix
269 optixGetMatrixFromSrt(trf0, trf1, trf2, srt);
270 }
271
272 // Returns the interpolated transformation matrix for a particular traversable handle and point in time.
273 static __forceinline__ __device__ void optixGetInterpolatedTransformationFromHandle(float4&
trf0,
274 float4&
trf1,
275 float4&
trf2,
276 const
OptixTraversableHandle handle,
277 const float
time,
278 const bool objectToWorld)
279 {
280 const OptixTransformType type = optixGetTransformTypeFromHandle(handle);
281
282 if(type == OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM || type ==
OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM)
283 {
284 if(type == OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM)
285 {
286 const OptixMatrixMotionTransform* transformData =
optixGetMatrixMotionTransformFromHandle(handle);
287 optixGetInterpolatedTransformation(trf0, trf1, trf2, transformData, time);
288 }
289 else
290 {
291 const OptixSRTMotionTransform* transformData = optixGetSRTMotionTransformFromHandle(handle);
292 optixGetInterpolatedTransformation(trf0, trf1, trf2, transformData, time);
293 }
294
295 if(!objectToWorld)
296 optixInvertMatrix(trf0, trf1, trf2);

```

```

297 }
298 else if(type == OPTIX_TRANSFORM_TYPE_INSTANCE || type == OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM)
299 {
300 const float4* transform;
301
302 if(type == OPTIX_TRANSFORM_TYPE_INSTANCE)
303 {
304 transform = (objectToWorld) ? optixGetInstanceTransformFromHandle(handle) :
305 optixGetInstanceInverseTransformFromHandle(handle);
306 }
307 else
308 {
309 const OptixStaticTransform* traversable = optixGetStaticTransformFromHandle(handle);
310 transform = (const float4*)((objectToWorld) ? traversable->transform :
traversable->invTransform);
311 }
312
313 trf0 = optixLoadReadOnlyAlign16(&transform[0]);
314 trf1 = optixLoadReadOnlyAlign16(&transform[1]);
315 trf2 = optixLoadReadOnlyAlign16(&transform[2]);
316 }
317 else
318 {
319 trf0 = {1.0f, 0.0f, 0.0f, 0.0f};
320 trf1 = {0.0f, 1.0f, 0.0f, 0.0f};
321 trf2 = {0.0f, 0.0f, 1.0f, 0.0f};
322 }
323 }
324
325 // Returns the world-to-object transformation matrix resulting from the current transform stack and
current ray time.
326 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float4& m0, float4& m1,
float4& m2)
327 {
328 const unsigned int size = optixGetTransformListSize();
329 const float time = optixGetRayTime();
330
331 #pragma unroll 1
332 for(unsigned int i = 0; i < size; ++i)
333 {
334 OptixTraversableHandle handle = optixGetTransformListHandle(i);
335
336 float4 trf0, trf1, trf2;
337 optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
false);
338
339 if(i == 0)
340 {
341 m0 = trf0;
342 m1 = trf1;
343 m2 = trf2;
344 }
345 else
346 {
347 // m := trf * m
348 float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
349 m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
350 m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
351 m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
352 }
353 }
354 }
355
356 // Returns the object-to-world transformation matrix resulting from the current transform stack and
current ray time.
357 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float4& m0, float4& m1,
float4& m2)

```

```

358 {
359 const int size = optixGetTransformListSize();
360 const float time = optixGetRayTime();
361
362 #pragma unroll 1
363 for(int i = size - 1; i >= 0; --i)
364 {
365 OptixTraversableHandle handle = optixGetTransformListHandle(i);
366
367 float4 trf0, trf1, trf2;
368 optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
true);
369
370 if(i == size - 1)
371 {
372 m0 = trf0;
373 m1 = trf1;
374 m2 = trf2;
375 }
376 else
377 {
378 // m := trf * m
379 float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
380 m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
381 m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
382 m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
383 }
384 }
385 }
386
387 // Multiplies the 3x4 matrix with rows m0, m1, m2 with the point p.
388 static __forceinline__ __device__ float3 optixTransformPoint(const float4& m0, const float4& m1, const
float4& m2, const float3& p)
389 {
390 float3 result;
391 result.x = m0.x * p.x + m0.y * p.y + m0.z * p.z + m0.w;
392 result.y = m1.x * p.x + m1.y * p.y + m1.z * p.z + m1.w;
393 result.z = m2.x * p.x + m2.y * p.y + m2.z * p.z + m2.w;
394 return result;
395 }
396
397 // Multiplies the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the vector v.
398 static __forceinline__ __device__ float3 optixTransformVector(const float4& m0, const float4& m1, const
float4& m2, const float3& v)
399 {
400 float3 result;
401 result.x = m0.x * v.x + m0.y * v.y + m0.z * v.z;
402 result.y = m1.x * v.x + m1.y * v.y + m1.z * v.z;
403 result.z = m2.x * v.x + m2.y * v.y + m2.z * v.z;
404 return result;
405 }
406
407 // Multiplies the transpose of the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the
normal n.
408 // Note that the given matrix is supposed to be the inverse of the actual transformation matrix.
409 static __forceinline__ __device__ float3 optixTransformNormal(const float4& m0, const float4& m1, const
float4& m2, const float3& n)
410 {
411 float3 result;
412 result.x = m0.x * n.x + m1.x * n.y + m2.x * n.z;
413 result.y = m0.y * n.x + m1.y * n.y + m2.y * n.z;
414 result.z = m0.z * n.x + m1.z * n.y + m2.z * n.z;
415 return result;
416 }
417
418 } // namespace optix_impl
419

```

```
420 #endif // OPTIX_OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H
```

## 8.5 optix\_micromap\_impl.h File Reference

### Namespaces

- namespace `optix_impl`

### Macros

- `#define OPTIX_MICROMAP_FUNC`
- `#define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline`
- `#define OPTIX_MICROMAP_FLOAT2_SUB(a, b) { a.x - b.x, a.y - b.y }`

### Functions

- `OPTIX_MICROMAP_INLINE_FUNC float optix_impl::_uint_as_float (unsigned int x)`
- `OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::extractEvenBits (unsigned int x)`
- `OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::prefixEor (unsigned int x)`
- `OPTIX_MICROMAP_INLINE_FUNC void optix_impl::index2dbary (unsigned int index, unsigned int &u, unsigned int &v, unsigned int &w)`
- `OPTIX_MICROMAP_INLINE_FUNC void optix_impl::micro2bary (unsigned int index, unsigned int subdivisionLevel, float2 &bary0, float2 &bary1, float2 &bary2)`
- `OPTIX_MICROMAP_INLINE_FUNC float2 optix_impl::base2micro (const float2 &baseBarycentrics, const float2 microVertexBaseBarycentrics[3])`

### 8.5.1 Detailed Description

OptiX micromap helper functions.

#### Author

NVIDIA Corporation

### 8.5.2 Macro Definition Documentation

#### 8.5.2.1 OPTIX\_MICROMAP\_FUNC

```
#define OPTIX_MICROMAP_FUNC
```

## 8.6 optix\_micromap\_impl.h

[Go to the documentation of this file.](#)

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2022 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
```

```

16 * contributors may be used to endorse or promote products derived from
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
31
38 #ifndef OPTIX_OPTIX_MICROMAP_IMPL_H
39 #define OPTIX_OPTIX_MICROMAP_IMPL_H
40
41 #ifndef OPTIX_MICROMAP_FUNC
42 #ifdef __CUDACC__
43 #define OPTIX_MICROMAP_FUNC __device__
44 #else
45 #define OPTIX_MICROMAP_FUNC
46 #endif
47 #endif
48
49 namespace optix_impl {
50
55 #define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline
56
57 #ifdef __CUDACC__
58 // the device implementation of __uint_as_float is declared in cuda_runtime.h
59 #else
60 // the host implementation of __uint_as_float
61 OPTIX_MICROMAP_INLINE_FUNC float __uint_as_float(unsigned int x)
62 {
63 union { float f; unsigned int i; } var;
64 var.i = x;
65 return var.f;
66 }
67 #endif
68
69 // Extract even bits
70 OPTIX_MICROMAP_INLINE_FUNC unsigned int extractEvenBits(unsigned int x)
71 {
72 x &= 0x55555555;
73 x = (x | (x >> 1)) & 0x33333333;
74 x = (x | (x >> 2)) & 0x0f0f0f0f;
75 x = (x | (x >> 4)) & 0x00ff00ff;
76 x = (x | (x >> 8)) & 0x0000ffff;
77 return x;
78 }
79
80
81 // Calculate exclusive prefix or (log(n) XOR's and SHF's)
82 OPTIX_MICROMAP_INLINE_FUNC unsigned int prefixEor(unsigned int x)
83 {
84 x ^= x >> 1;
85 x ^= x >> 2;
86 x ^= x >> 4;
87 x ^= x >> 8;
88 return x;
89 }
90
91 // Convert distance along the curve to discrete barycentrics
92 OPTIX_MICROMAP_INLINE_FUNC void index2dbary(unsigned int index, unsigned int& u, unsigned int& v, unsigned

```

```

int& w)
93 {
94 unsigned int b0 = extractEvenBits(index);
95 unsigned int b1 = extractEvenBits(index » 1);
96
97 unsigned int fx = prefixEor(b0);
98 unsigned int fy = prefixEor(b0 & ~b1);
99
100 unsigned int t = fy ^ b1;
101
102 u = (fx & ~t) | (b0 & ~t) | (~b0 & ~fx & t);
103 v = fy ^ b0;
104 w = (~fx & ~t) | (b0 & ~t) | (~b0 & fx & t);
105 }
106
107 // Compute barycentrics of a sub or micro triangle wrt a base triangle. The order of the returned
108 // bary0, bary1, bary2 matters and allows for using this function for sub triangles and the
109 // conversion from sub triangle to base triangle barycentric space
110 OPTIX_MICROMAP_INLINE_FUNC void micro2bary(unsigned int index, unsigned int subdivisionLevel, float2&
bary0, float2& bary1, float2& bary2)
111 {
112 if(subdivisionLevel == 0)
113 {
114 bary0 = { 0, 0 };
115 bary1 = { 1, 0 };
116 bary2 = { 0, 1 };
117 return;
118 }
119
120 unsigned int iu, iv, iw;
121 index2dbary(index, iu, iv, iw);
122
123 // we need to only look at "level" bits
124 iu = iu & ((1 « subdivisionLevel) - 1);
125 iv = iv & ((1 « subdivisionLevel) - 1);
126 iw = iw & ((1 « subdivisionLevel) - 1);
127
128 int yFlipped = (iu & 1) ^ (iv & 1) ^ (iw & 1) ^ 1;
129
130 int xFlipped = ((0x8888888888888888ull ^ 0xf000f000f000f000ull ^ 0xffff000000000000ull) » index) & 1;
131 xFlipped ^= ((0x8888888888888888ull ^ 0xf000f000f000f000ull ^ 0xffff000000000000ull) » (index »
6)) & 1;
132
133 const float levelScale = __uint_as_float((127u - subdivisionLevel) « 23);
134
135 // scale the barycentric coordinate to the global space/scale
136 float du = 1.f * levelScale;
137 float dv = 1.f * levelScale;
138
139 // scale the barycentric coordinate to the global space/scale
140 float u = (float)iu * levelScale;
141 float v = (float)iv * levelScale;
142
143 // c d
144 // x-----x
145 // / \ /
146 // / \ /
147 // x-----x
148 // a b
149 //
150 // !xFlipped && !yFlipped: abc
151 // !xFlipped && yFlipped: cdb
152 // xFlipped && !yFlipped: bac
153 // xFlipped && yFlipped: dcb
154
155 bary0 = { u + xFlipped * du, v + yFlipped * dv };
156 bary1 = { u + (1-xFlipped) * du, v + yFlipped * dv };

```

```

157 bary2 = { u + yFlipped * du , v + (1-yFlipped) * dv };
158 }
159
160 // avoid any conflicts due to multiple definitions
161 #define OPTIX_MICROMAP_FLOAT2_SUB(a,b) { a.x - b.x, a.y - b.y }
162
163 // Compute barycentrics for micro triangle from base barycentrics
164 OPTIX_MICROMAP_INLINE_FUNC float2 base2micro(const float2& baseBarycentrics, const float2
microVertexBaseBarycentrics[3])
165 {
166 float2 baryV0P = OPTIX_MICROMAP_FLOAT2_SUB(baseBarycentrics, microVertexBaseBarycentrics[0]);
167 float2 baryV0V1 = OPTIX_MICROMAP_FLOAT2_SUB(microVertexBaseBarycentrics[1],
microVertexBaseBarycentrics[0]);
168 float2 baryV0V2 = OPTIX_MICROMAP_FLOAT2_SUB(microVertexBaseBarycentrics[2],
microVertexBaseBarycentrics[0]);
169
170 float rdetA = 1.f / (baryV0V1.x * baryV0V2.y - baryV0V1.y * baryV0V2.x);
171 float4 A = { baryV0V2.y, -baryV0V2.x, -baryV0V1.y, baryV0V1.x };
172
173 float2 localUV;
174 localUV.x = rdetA * (baryV0P.x * A.x + baryV0P.y * A.y);
175 localUV.y = rdetA * (baryV0P.x * A.z + baryV0P.y * A.w);
176
177 return localUV;
178 }
179 #undef OPTIX_MICROMAP_FLOAT2_SUB
180 // end group optix_utilities
181
182
183 } // namespace optix_impl
184
185 #endif // OPTIX_OPTIX_MICROMAP_IMPL_H

```

## 8.7 optix.h File Reference

### Macros

- #define `OPTIX_VERSION` 80100

### 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 80100
```

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 * SPDX-FileCopyrightText: Copyright (c) 2009 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
4 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
5 *
6 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
7 * property and proprietary rights in and to this material, related
8 * documentation and any modifications thereto. Any use, reproduction,
9 * disclosure or distribution of this material and related documentation
10 * without an express license agreement from NVIDIA CORPORATION or
11 * its affiliates is strictly prohibited.
12 */
19
20 #ifndef OPTIX_OPTIX_H
21 #define OPTIX_OPTIX_H
22
23 #define OPTIX_VERSION 80100
24
25
26 #ifdef __CUDACC__
27 #include "optix_device.h"
28 #else
29 #include "optix_host.h"
30 #endif
31
32 #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 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
16 * contributors may be used to endorse or promote products derived from
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
31
32
33
34
35 #ifndef OPTIX_DENOISER_TILING_H
36 #define OPTIX_DENOISER_TILING_H
37
38 #include <optix.h>
39
40 #include <algorithm>
41 #include <vector>
42
43 #ifdef __cplusplus
44 extern "C" {
45 #endif
46
47 struct OptixUtilDenoiserImageTile
48 {
49 // input tile image
50 OptixImage2D input;
51
52 // output tile image
53 OptixImage2D output;
54
55 // overlap offsets, parameters for #optixUtilDenoiserInvoke
56 unsigned int inputOffsetX;
57 unsigned int inputOffsetY;
58 };
59
60 inline OptixResult optixUtilGetPixelStride(const OptixImage2D& image, unsigned int& pixelStrideInBytes)
61 {
62 pixelStrideInBytes = image.pixelStrideInBytes;
63 if(pixelStrideInBytes == 0)
64 {
65 switch(image.format)
66 {
67 case OPTIX_PIXEL_FORMAT_HALF1:
68 pixelStrideInBytes = 1 * sizeof(short);
69 break;
70 case OPTIX_PIXEL_FORMAT_HALF2:

```

```

86 pixelStrideInBytes = 2 * sizeof(short);
87 break;
88 case OPTIX_PIXEL_FORMAT_HALF3:
89 pixelStrideInBytes = 3 * sizeof(short);
90 break;
91 case OPTIX_PIXEL_FORMAT_HALF4:
92 pixelStrideInBytes = 4 * sizeof(short);
93 break;
94 case OPTIX_PIXEL_FORMAT_FLOAT1:
95 pixelStrideInBytes = 1 * sizeof(float);
96 break;
97 case OPTIX_PIXEL_FORMAT_FLOAT2:
98 pixelStrideInBytes = 2 * sizeof(float);
99 break;
100 case OPTIX_PIXEL_FORMAT_FLOAT3:
101 pixelStrideInBytes = 3 * sizeof(float);
102 break;
103 case OPTIX_PIXEL_FORMAT_FLOAT4:
104 pixelStrideInBytes = 4 * sizeof(float);
105 break;
106 case OPTIX_PIXEL_FORMAT_UCHAR3:
107 pixelStrideInBytes = 3 * sizeof(char);
108 break;
109 case OPTIX_PIXEL_FORMAT_UCHAR4:
110 pixelStrideInBytes = 4 * sizeof(char);
111 break;
112 case OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER:
113 return OPTIX_ERROR_INVALID_VALUE;
114 break;
115 }
116 }
117 return OPTIX_SUCCESS;
118 }
119
120 inline OptixResult optixUtilDenoiserSplitImage(
121 const OptixImage2D& input,
122 const OptixImage2D& output,
123 unsigned int overlapWindowSizeInPixels,
124 unsigned int tileWidth,
125 unsigned int tileHeight,
126 std::vector<OptixUtilDenoiserImageTile>& tiles)
127 {
128 if(tileWidth == 0 || tileHeight == 0)
129 return OPTIX_ERROR_INVALID_VALUE;
130
131 unsigned int inPixelStride, outPixelStride;
132 if(const OptixResult res = optixUtilGetPixelStride(input, inPixelStride))
133 return res;
134 if(const OptixResult res = optixUtilGetPixelStride(output, outPixelStride))
135 return res;
136
137 int inp_w = std::min(tileWidth + 2 * overlapWindowSizeInPixels, input.width);
138 int inp_h = std::min(tileHeight + 2 * overlapWindowSizeInPixels, input.height);
139 int inp_y = 0, copied_y = 0;
140
141 int upscaleX = output.width / input.width;
142 int upscaleY = output.height / input.height;
143
144 do
145 {
146 int inputOffsetY = inp_y == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_h -
147 ((int)input.height - inp_y));
148 int copy_y = inp_y == 0 ? std::min(input.height, tileHeight + overlapWindowSizeInPixels) :
149 std::min(tileHeight, input.height - copied_y);
150
151 int inp_x = 0, copied_x = 0;
152 }
153 do

```

```

161 {
162 int inputOffsetX = inp_x == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_w -
((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 OptixUtilDenoiserImageTile tile;
167 tile.input.data = input.data + (size_t)(inp_y - inputOffsetY) *
input.rowStrideInBytes
168 + (size_t)(inp_x - inputOffsetX) * inPixelStride;
169 tile.input.width = inp_w;
170 tile.input.height = inp_h;
171 tile.input.rowStrideInBytes = input.rowStrideInBytes;
172 tile.input.pixelStrideInBytes = input.pixelStrideInBytes;
173 tile.input.format = input.format;
174
175 tile.output.data = output.data + (size_t)(upscaleY * inp_y) *
output.rowStrideInBytes
176 + (size_t)(upscaleX * inp_x) * outPixelStride;
177 tile.output.width = upscaleX * copy_x;
178 tile.output.height = upscaleY * copy_y;
179 tile.output.rowStrideInBytes = output.rowStrideInBytes;
180 tile.output.pixelStrideInBytes = output.pixelStrideInBytes;
181 tile.output.format = output.format;
182
183 tile.inputOffsetX = inputOffsetX;
184 tile.inputOffsetY = inputOffsetY;
185
186 tiles.push_back(tile);
187
188 inp_x += inp_x == 0 ? tileWidth + overlapWindowSizeInPixels : tileWidth;
189 copied_x += copy_x;
190 } while(inp_x < static_cast<int>(input.width));
191
192 inp_y += inp_y == 0 ? tileHeight + overlapWindowSizeInPixels : tileHeight;
193 copied_y += copy_y;
194 } while(inp_y < static_cast<int>(input.height));
195
196 return OPTIX_SUCCESS;
197 }
198
199
200
201
202
203
204
205 inline OptixResult optixUtilDenoiserInvokeTiled(
206 OptixDenoiser denoiser,
207 CUstream stream,
208 const OptixDenoiserParams* params,
209 CUdeviceptr denoiserState,
210 size_t denoiserStateSizeInBytes,
211 const OptixDenoiserGuideLayer* guideLayer,
212 const OptixDenoiserLayer* layers,
213 unsigned int numLayers,
214 CUdeviceptr scratch,
215 size_t scratchSizeInBytes,
216 unsigned int overlapWindowSizeInPixels,
217 unsigned int tileWidth,
218 unsigned int tileHeight)
219 {
220 if(!guideLayer || !layers)
221 return OPTIX_ERROR_INVALID_VALUE;
222
223 const unsigned int upscale = numLayers > 0 && layers[0].previousOutput.width == 2 *
layers[0].input.width ? 2 : 1;
224
225 std::vector<std::vector<OptixUtilDenoiserImageTile>> tiles(numLayers);
226 std::vector<std::vector<OptixUtilDenoiserImageTile>> prevTiles(numLayers);
227 for(unsigned int l = 0; l < numLayers; l++)

```

```

248 {
249 if(const OptixResult res = optixUtilDenoiserSplitImage(layers[l].input, layers[l].output,
250 overlapWindowSizeInPixels,
251 tileWidth, tileHeight, tiles[l]))
252 return res;
253
254 if(layers[l].previousOutput.data)
255 {
256 OptixImage2D dummyOutput = layers[l].previousOutput;
257 if(const OptixResult res = optixUtilDenoiserSplitImage(layers[l].previousOutput, dummyOutput,
258 upscale * overlapWindowSizeInPixels,
259 upscale * tileWidth, upscale * tileHeight,
prevTiles[l]))
260 return res;
261 }
262 }
263
264 std::vector<OptixUtilDenoiserImageTile> albedoTiles;
265 if(guidelayer->albedo.data)
266 {
267 OptixImage2D dummyOutput = guideline->albedo;
268 if(const OptixResult res = optixUtilDenoiserSplitImage(guidelayer->albedo, dummyOutput,
269 overlapWindowSizeInPixels,
270 tileWidth, tileHeight, albedoTiles))
271 return res;
272 }
273
274 std::vector<OptixUtilDenoiserImageTile> normalTiles;
275 if(guidelayer->normal.data)
276 {
277 OptixImage2D dummyOutput = guideline->normal;
278 if(const OptixResult res = optixUtilDenoiserSplitImage(guidelayer->normal, dummyOutput,
279 overlapWindowSizeInPixels,
280 tileWidth, tileHeight, normalTiles))
281 return res;
282 }
283
284 std::vector<OptixUtilDenoiserImageTile> flowTiles;
285 if(guidelayer->flow.data)
286 {
287 OptixImage2D dummyOutput = guideline->flow;
288 if(const OptixResult res = optixUtilDenoiserSplitImage(guidelayer->flow, dummyOutput,
289 overlapWindowSizeInPixels,
290 tileWidth, tileHeight, flowTiles))
291 return res;
292 }
293
294 std::vector<OptixUtilDenoiserImageTile> flowTrustTiles;
295 if(guidelayer->flowTrustworthiness.data)
296 {
297 OptixImage2D dummyOutput = guideline->flowTrustworthiness;
298 if(const OptixResult res = optixUtilDenoiserSplitImage(guidelayer->flowTrustworthiness,
dummyOutput,
299 overlapWindowSizeInPixels,
300 tileWidth, tileHeight, flowTrustTiles))
301 return res;
302 }
303
304 std::vector<OptixUtilDenoiserImageTile> internalGuideLayerTiles;
305 if(guidelayer->previousOutputInternalGuideLayer.data && guideline->outputInternalGuideLayer.data)
306 {
307 if(const OptixResult res =
optixUtilDenoiserSplitImage(guidelayer->previousOutputInternalGuideLayer,
308 guideline->outputInternalGuideLayer,
309 upscale * overlapWindowSizeInPixels,
310 upscale * tileWidth, upscale * tileHeight,
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
358 #ifdef __cplusplus
359 }
360 #endif
361 #endif
362
363 #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 optixTraverse (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)`
  - `template<typename... Payload>`  
`static __forceinline__ __device__ void optixTraverse (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 optixReorder (unsigned int coherenceHint, unsigned int numCoherenceHintBitsFromLSB)`
  - `static __forceinline__ __device__ void optixReorder ()`
  - `template<typename... Payload>`  
`static __forceinline__ __device__ void optixInvoke (Payload &... payload)`
  - `template<typename... Payload>`  
`static __forceinline__ __device__ void optixInvoke (OptixPayloadTypeID type, Payload &... payload)`
  - `template<typename... RegAttributes>`  
`static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int SBTOffset, unsigned int SBTStride, unsigned int instIdx, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)`
  - `template<typename... RegAttributes>`  
`static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int SBTOffset, unsigned int SBTStride, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)`
  - `template<typename... RegAttributes>`  
`static __forceinline__ __device__ void optixMakeHitObjectWithRecord (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int sbtRecordIndex, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)`
  - `static __forceinline__ __device__ void optixMakeMissHitObject (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime)`
  - `static __forceinline__ __device__ void optixMakeNopHitObject ()`
  - `static __forceinline__ __device__ bool optixHitObjectIsHit ()`
  - `static __forceinline__ __device__ bool optixHitObjectIsMiss ()`
  - `static __forceinline__ __device__ bool optixHitObjectIsNop ()`
  - `static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex ()`
  - `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\_\_ 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 optixHitObjectGetWorldRayOrigin ()
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixGetWorldRayDirection ()
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixHitObjectGetWorldRayDirection ()
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixGetObjectRayOrigin ()
- static \_\_forceinline\_\_ \_\_device\_\_ float3 optixGetObjectRayDirection ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetRayTmin ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixHitObjectGetRayTmin ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetRayTmax ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixHitObjectGetRayTmax ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixGetRayTime ()
- static \_\_forceinline\_\_ \_\_device\_\_ float optixHitObjectGetRayTime ()
- 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__ unsigned int optixHitObjectGetTransformListSize ()`
- static `__forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle (unsigned int index)`
- static `__forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle (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 `__device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle (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__ unsigned int optixHitObjectGetAttribute_0` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7` ()
- static `__forceinline__ __device__ void optixTerminateRay` ()
- static `__forceinline__ __device__ void optixIgnoreIntersection` ()
- static `__forceinline__ __device__ unsigned int optixGetPrimitiveIndex` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex` ()
- static `__forceinline__ __device__ unsigned int optixGetSbtGASIndex` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex` ()
- static `__forceinline__ __device__ unsigned int optixGetInstanceId` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetInstanceId` ()
- static `__forceinline__ __device__ unsigned int optixGetInstanceIndex` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex` ()
- static `__forceinline__ __device__ unsigned int optixGetHitKind` ()
- static `__forceinline__ __device__ unsigned int optixHitObjectGetHitKind` ()
- 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__ CUdeviceptr optixHitObjectGetSbtDataPointer ()`
- 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__ 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 * SPDX-FileCopyrightText: Copyright (c) 2010 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
4 *
5 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
6 * property and proprietary rights in and to this material, related
7 * documentation and any modifications thereto. Any use, reproduction,
8 * disclosure or distribution of this material and related documentation
9 * without an express license agreement from NVIDIA CORPORATION or
10 * its affiliates is strictly prohibited.
11 */
12
13
14
15
16
17
18 #ifndef OPTIX_OPTIX_DEVICE_H
19 #define OPTIX_OPTIX_DEVICE_H
20
21 #if defined(__cplusplus) && (__cplusplus < 201103L) && !defined(_WIN32)
22 #error Device code for OptiX requires at least C++11. Consider adding "--std c++11" to the nvcc
23 command-line.
24 #endif
25 #include "optix_types.h"
26
27
28
29
30
31 template <typename... Payload>
32 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
33 float3 rayOrigin,
34 float3 rayDirection,
35 float tmin,
36 float tmax,
37 float rayTime,
38 OptixVisibilityMask visibilityMask,
39 unsigned int rayFlags,
40 unsigned int SBTOffset,
41 unsigned int SBTstride,
```

```

62 unsigned int missSBTIndex,
63 Payload&... payload);
64
65 template <typename... Payload>
66 static __forceinline__ __device__ void optixTraverse(OptixTraversableHandle handle,
67 float3 rayOrigin,
68 float3 rayDirection,
69 float tmin,
70 float tmax,
71 float rayTime,
72 OptixVisibilityMask visibilityMask,
73 unsigned int rayFlags,
74 unsigned int SBTOffset,
75 unsigned int SBTstride,
76 unsigned int missSBTIndex,
77 Payload&... payload);
78
79 template <typename... Payload>
80 static __forceinline__ __device__ void optixTrace(OptixPayloadTypeID type,
81 OptixTraversableHandle handle,
82 float3 rayOrigin,
83 float3 rayDirection,
84 float tmin,
85 float tmax,
86 float rayTime,
87 OptixVisibilityMask visibilityMask,
88 unsigned int rayFlags,
89 unsigned int SBTOffset,
90 unsigned int SBTstride,
91 unsigned int missSBTIndex,
92 Payload&... payload);
93
94 template <typename... Payload>
95 static __forceinline__ __device__ void optixTraverse(OptixPayloadTypeID type,
96 OptixTraversableHandle handle,
97 float3 rayOrigin,
98 float3 rayDirection,
99 float tmin,
100 float tmax,
101 float rayTime,
102 OptixVisibilityMask visibilityMask,
103 unsigned int rayFlags,
104 unsigned int SBTOffset,
105 unsigned int SBTstride,
106 unsigned int missSBTIndex,
107 Payload&... payload);
108
109 static __forceinline__ __device__ void optixReorder(unsigned int coherenceHint, unsigned int
numCoherenceHintBitsFromLSB);
110
111 static __forceinline__ __device__ void optixReorder();
112
113 template <typename... Payload>
114 static __forceinline__ __device__ void optixInvoke(Payload&... payload);
115
116 template <typename... Payload>
117 static __forceinline__ __device__ void optixInvoke(OptixPayloadTypeID type, Payload&... payload);
118
119 template <typename... RegAttributes>
120 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle handle,
121 float3 rayOrigin,
122 float3 rayDirection,
123 float tmin,
124 float tmax,
125 float rayTime,
126 unsigned int SBTOffset,
127 unsigned int SBTstride,
128 unsigned int missSBTIndex,
129 Payload&... payload);

```

```

232 unsigned int instIdx,
233 unsigned int sbtGASIdx,
234 unsigned int primIdx,
235 unsigned int hitKind,
236 RegAttributes... regAttributes);
237
260 template <typename... RegAttributes>
261 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle handle,
262 float3 rayOrigin,
263 float3 rayDirection,
264 float tmin,
265 float tmax,
266 float rayTime,
267 unsigned int sbtOffset,
268 unsigned int sbtStride,
269 unsigned int instIdx,
270 const OptixTraversableHandle* transforms,
271 unsigned int numTransforms,
272 unsigned int sbtGASIdx,
273 unsigned int primIdx,
274 unsigned int hitKind,
275 RegAttributes... regAttributes);
276
297 template <typename... RegAttributes>
298 static __forceinline__ __device__ void optixMakeHitObjectWithRecord(OptixTraversableHandle handle,
299 float3 rayOrigin,
300 float3 rayDirection,
301 float tmin,
302 float tmax,
303 float rayTime,
304 unsigned int sbtRecordIndex,
305 unsigned int instIdx,
306 const OptixTraversableHandle* transforms,
307 unsigned int numTransforms,
308 unsigned int sbtGASIdx,
309 unsigned int primIdx,
310 unsigned int hitKind,
311 RegAttributes... regAttributes);
312
325 static __forceinline__ __device__ void optixMakeMissHitObject(unsigned int missSBTIndex,
326 float3 rayOrigin,
327 float3 rayDirection,
328 float tmin,
329 float tmax,
330 float rayTime);
331
339 static __forceinline__ __device__ void optixMakeNopHitObject();
340
344 static __forceinline__ __device__ bool optixHitObjectIsHit();
345
349 static __forceinline__ __device__ bool optixHitObjectIsMiss();
350
356 static __forceinline__ __device__ bool optixHitObjectIsNop();
357
364 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex();
365
371 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p);
372 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p);
373 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p);
374 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p);
375 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p);
376 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p);
377 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p);
378 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p);
379 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p);
380 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p);
381 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p);

```

```

382 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p);
383 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p);
384 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p);
385 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p);
386 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p);
387 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p);
388 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p);
389 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p);
390 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p);
391 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p);
392 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p);
393 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p);
394 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p);
395 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p);
396 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p);
397 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p);
398 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p);
399 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p);
400 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p);
401 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p);
402 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p);
403
409 static __forceinline__ __device__ unsigned int optixGetPayload_0();
410 static __forceinline__ __device__ unsigned int optixGetPayload_1();
411 static __forceinline__ __device__ unsigned int optixGetPayload_2();
412 static __forceinline__ __device__ unsigned int optixGetPayload_3();
413 static __forceinline__ __device__ unsigned int optixGetPayload_4();
414 static __forceinline__ __device__ unsigned int optixGetPayload_5();
415 static __forceinline__ __device__ unsigned int optixGetPayload_6();
416 static __forceinline__ __device__ unsigned int optixGetPayload_7();
417 static __forceinline__ __device__ unsigned int optixGetPayload_8();
418 static __forceinline__ __device__ unsigned int optixGetPayload_9();
419 static __forceinline__ __device__ unsigned int optixGetPayload_10();
420 static __forceinline__ __device__ unsigned int optixGetPayload_11();
421 static __forceinline__ __device__ unsigned int optixGetPayload_12();
422 static __forceinline__ __device__ unsigned int optixGetPayload_13();
423 static __forceinline__ __device__ unsigned int optixGetPayload_14();
424 static __forceinline__ __device__ unsigned int optixGetPayload_15();
425 static __forceinline__ __device__ unsigned int optixGetPayload_16();
426 static __forceinline__ __device__ unsigned int optixGetPayload_17();
427 static __forceinline__ __device__ unsigned int optixGetPayload_18();
428 static __forceinline__ __device__ unsigned int optixGetPayload_19();
429 static __forceinline__ __device__ unsigned int optixGetPayload_20();
430 static __forceinline__ __device__ unsigned int optixGetPayload_21();
431 static __forceinline__ __device__ unsigned int optixGetPayload_22();
432 static __forceinline__ __device__ unsigned int optixGetPayload_23();
433 static __forceinline__ __device__ unsigned int optixGetPayload_24();
434 static __forceinline__ __device__ unsigned int optixGetPayload_25();
435 static __forceinline__ __device__ unsigned int optixGetPayload_26();
436 static __forceinline__ __device__ unsigned int optixGetPayload_27();
437 static __forceinline__ __device__ unsigned int optixGetPayload_28();
438 static __forceinline__ __device__ unsigned int optixGetPayload_29();
439 static __forceinline__ __device__ unsigned int optixGetPayload_30();
440 static __forceinline__ __device__ unsigned int optixGetPayload_31();
441
450 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int typeMask);
451
455 static __forceinline__ __device__ unsigned int optixUndefinedValue();
456
463 static __forceinline__ __device__ float3 optixGetWorldRayOrigin();
464
471 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin();
472
479 static __forceinline__ __device__ float3 optixGetWorldRayDirection();
480
487 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection();
488

```

```
492 static __forceinline__ __device__ float3 optixGetObjectRayOrigin();
493
497 static __forceinline__ __device__ float3 optixGetObjectRayDirection();
498
502 static __forceinline__ __device__ float optixGetRayTmin();
503
510 static __forceinline__ __device__ float optixHitObjectGetRayTmin();
511
520 static __forceinline__ __device__ float optixGetRayTmax();
521
530 static __forceinline__ __device__ float optixHitObjectGetRayTmax();
531
537 static __forceinline__ __device__ float optixGetRayTime();
538
545 static __forceinline__ __device__ float optixHitObjectGetRayTime();
546
550 static __forceinline__ __device__ unsigned int optixGetRayFlags();
551
555 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask();
556
563 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias, unsigned int instIdx);
564
575 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas, unsigned
int primIdx, unsigned int sbtGASIndex, float time, float3 data[3]);
576
581 static __forceinline__ __device__ void optixGetMicroTriangleVertexData(float3 data[3]);
582
587 static __forceinline__ __device__ void optixGetMicroTriangleBarycentricsData(float2 data[3]);
588
601 static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2]);
602
615 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3]);
616
629 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4]);
630
643 static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas, unsigned
int primIdx, unsigned int sbtGASIndex, float time, float4 data[4]);
644
657 static __forceinline__ __device__ void optixGetCubicBezierVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4]);
658
671 static __forceinline__ __device__ void optixGetRibbonVertexData(OptixTraversableHandle gas, unsigned int
primIdx, unsigned int sbtGASIndex, float time, float4 data[3]);
672
676 static __forceinline__ __device__ float3 optixGetRibbonNormal(OptixTraversableHandle gas, unsigned int
primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters);
677
690 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas, unsigned int
primIdx, unsigned int sbtGASIndex, float time, float4 data[1]);
691
696 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle();
697
701 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle gas);
702
706 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle gas);
707
711 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle gas);
712
719 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12]);
720
727 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12]);
728
735 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point);
```

```

736
743 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec);
744
751 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal);
752
759 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point);
760
767 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec);
768
775 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal);
776
780 static __forceinline__ __device__ unsigned int optixGetTransformListSize();
781
790 static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize();
791
795 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int index);
796
805 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle(unsigned
int index);
806
810 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle);
811
817 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle);
818
824 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle);
825
831 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle);
832
838 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle);
839
845 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle(OptixTraversableHandle handle);
846
852 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle);
853
859 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle);
860
866 static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle(OptixTraversableHandle
handle);
890 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind);
891
897 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0);
898
904 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1);
905
911 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1, unsigned int a2);
912
918 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
unsigned int hitKind,
919 unsigned int a0,
920 unsigned int a1,
921 unsigned int a2,
922 unsigned int a3);
923
924
930 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
unsigned int hitKind,
931 unsigned int a0,
932

```

```

933 unsigned int a1,
934 unsigned int a2,
935 unsigned int a3,
936 unsigned int a4);
937
943 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
944 unsigned int hitKind,
945 unsigned int a0,
946 unsigned int a1,
947 unsigned int a2,
948 unsigned int a3,
949 unsigned int a4,
950 unsigned int a5);
951
957 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
958 unsigned int hitKind,
959 unsigned int a0,
960 unsigned int a1,
961 unsigned int a2,
962 unsigned int a3,
963 unsigned int a4,
964 unsigned int a5,
965 unsigned int a6);
966
972 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
973 unsigned int hitKind,
974 unsigned int a0,
975 unsigned int a1,
976 unsigned int a2,
977 unsigned int a3,
978 unsigned int a4,
979 unsigned int a5,
980 unsigned int a6,
981 unsigned int a7);
982
987 static __forceinline__ __device__ unsigned int optixGetAttribute_0();
988 static __forceinline__ __device__ unsigned int optixGetAttribute_1();
989 static __forceinline__ __device__ unsigned int optixGetAttribute_2();
990 static __forceinline__ __device__ unsigned int optixGetAttribute_3();
991 static __forceinline__ __device__ unsigned int optixGetAttribute_4();
992 static __forceinline__ __device__ unsigned int optixGetAttribute_5();
993 static __forceinline__ __device__ unsigned int optixGetAttribute_6();
994 static __forceinline__ __device__ unsigned int optixGetAttribute_7();
995
996
1004 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0();
1005 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1();
1006 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2();
1007 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3();
1008 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4();
1009 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5();
1010 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6();
1011 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7();
1012
1016 static __forceinline__ __device__ void optixTerminateRay();
1017
1022 static __forceinline__ __device__ void optixIgnoreIntersection();
1023
1024
1040 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex();
1041
1049 static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex();
1050
1058 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex();
1059
1068 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex();
1069

```

```

1070
1083 static __forceinline__ __device__ unsigned int optixGetInstanceId();
1084
1093 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId();
1094
1104 static __forceinline__ __device__ unsigned int optixGetInstanceIndex();
1105
1114 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex();
1115
1123 static __forceinline__ __device__ unsigned int optixGetHitKind();
1124
1132 static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind();
1133
1137 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind);
1138
1142 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind);
1143
1147 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind);
1148
1152 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType();
1153
1157 static __forceinline__ __device__ bool optixIsFrontFaceHit();
1158
1162 static __forceinline__ __device__ bool optixIsBackFaceHit();
1163
1167 static __forceinline__ __device__ bool optixIsTriangleHit();
1168
1172 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit();
1173
1177 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit();
1178
1182 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit();
1183
1187 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleFrontFaceHit();
1188
1192 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleBackFaceHit();
1193
1200 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics();
1201
1206 static __forceinline__ __device__ float optixGetCurveParameter();
1207
1213 static __forceinline__ __device__ float2 optixGetRibbonParameters();
1214
1221 static __forceinline__ __device__ uint3 optixGetLaunchIndex();
1222
1227 static __forceinline__ __device__ uint3 optixGetLaunchDimensions();
1228
1236 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer();
1237
1244 static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer();
1245
1260 static __forceinline__ __device__ void optixThrowException(int exceptionCode);
1261
1267 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0);
1268
1274 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
unsigned int exceptionDetail0,
unsigned int exceptionDetail1);
1275
1276
1277
1283 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
unsigned int exceptionDetail0,
unsigned int exceptionDetail1,
unsigned int exceptionDetail2);
1284
1285
1286
1287
1293 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
unsigned int exceptionDetail0,
1294

```

```

1295 unsigned int exceptionDetail1,
1296 unsigned int exceptionDetail2,
1297 unsigned int exceptionDetail3);
1298
1304 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1305 unsigned int exceptionDetail0,
1306 unsigned int exceptionDetail1,
1307 unsigned int exceptionDetail2,
1308 unsigned int exceptionDetail3,
1309 unsigned int exceptionDetail4);
1310
1316 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1317 unsigned int exceptionDetail0,
1318 unsigned int exceptionDetail1,
1319 unsigned int exceptionDetail2,
1320 unsigned int exceptionDetail3,
1321 unsigned int exceptionDetail4,
1322 unsigned int exceptionDetail5);
1323
1330 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1331 unsigned int exceptionDetail0,
1332 unsigned int exceptionDetail1,
1333 unsigned int exceptionDetail2,
1334 unsigned int exceptionDetail3,
1335 unsigned int exceptionDetail4,
1336 unsigned int exceptionDetail5,
1337 unsigned int exceptionDetail6);
1338
1344 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1345 unsigned int exceptionDetail0,
1346 unsigned int exceptionDetail1,
1347 unsigned int exceptionDetail2,
1348 unsigned int exceptionDetail3,
1349 unsigned int exceptionDetail4,
1350 unsigned int exceptionDetail5,
1351 unsigned int exceptionDetail6,
1352 unsigned int exceptionDetail7);
1353
1357 static __forceinline__ __device__ int optixGetExceptionCode();
1358
1365 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0();
1366
1372 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1();
1373
1379 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2();
1380
1386 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3();
1387
1393 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4();
1394
1400 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5();
1401
1407 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6();
1408
1414 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7();
1415
1416
1427 static __forceinline__ __device__ char* optixGetExceptionLineInfo();
1428
1452 template <typename ReturnT, typename... ArgTypes>
1453 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args);
1454
1455
1478 template <typename ReturnT, typename... ArgTypes>
1479 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes...
args);
1480

```

```

1481
1546 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int
texInfo, float x, float y, unsigned int* singleMipLevel);
1547
1559 static __forceinline__ __device__ uint4
1560 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool
coarse, unsigned int* singleMipLevel);
1561
1576 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
1577 unsigned int texInfo,
1578 float x,
1579 float y,
1580 float dPdx_x,
1581 float dPdx_y,
1582 float dPdy_x,
1583 float dPdy_y,
1584 bool coarse,
1585 unsigned int* singleMipLevel);
1586 // end group optix_device_api
1588
1589 #define __OPTIX_INCLUDE_INTERNAL_HEADERS__
1590
1591 #include "internal/optix_device_impl.h"
1592
1593 #endif // OPTIX_DEVICE_H

```

## 8.13 optix\_function\_table.h File Reference

### Classes

- struct [OptixFunctionTable](#)

### Macros

- #define `OPTIX_ABI_VERSION` 93
- #define `OPTIX_CONCATENATE_ABI_VERSION(prefix, macro) OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro)`
- #define `OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro) prefix ## _ ## macro`
- #define `OPTIX_FUNCTION_TABLE_SYMBOL OPTIX_CONCATENATE_ABI_VERSION(g_optixFunctionTable, OPTIX_ABI_VERSION)`

### 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 93
```

The OptiX ABI version.

## 8.14 optix\_function\_table.h

Go to the documentation of this file.

```

1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
4 *
5 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
6 * property and proprietary rights in and to this material, related
7 * documentation and any modifications thereto. Any use, reproduction,
8 * disclosure or distribution of this material and related documentation
9 * without an express license agreement from NVIDIA CORPORATION or
10 * its affiliates is strictly prohibited.
11 */
12
13 #ifndef OPTIX_OPTIX_FUNCTION_TABLE_H
14 #define OPTIX_OPTIX_FUNCTION_TABLE_H
15
16 #define OPTIX_ABI_VERSION 93
17
18 #ifndef OPTIX_DEFINE_ABI_VERSION_ONLY
19 #include "optix_types.h"
20 #endif
21
22 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
23 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver types must be defined through other
24 // means before including optix headers.
25 #include <cuda.h>
26 #endif
27
28 #ifdef __cplusplus
29 extern "C" {
30 #endif
31
32 typedef struct OptixFunctionTable
33 {
34 /*@ {
35
36 const char* (*optixGetErrorName)(OptixResult result);
37
38 const char* (*optixGetErrorString)(OptixResult result);
39 /*@ }
40 /*@ {
41
42 OptixResult (*optixDeviceContextCreate)(CUcontext fromContext, const OptixDeviceContextOptions*
43 options, OptixDeviceContext* context);
44
45 OptixResult (*optixDeviceContextDestroy)(OptixDeviceContext context);
46
47 OptixResult (*optixDeviceContextGetProperty)(OptixDeviceContext context, OptixDeviceProperty
48 property, void* value, size_t sizeInBytes);
49
50 OptixResult (*optixDeviceContextSetLogCallback)(OptixDeviceContext context,
51 OptixLogCallback callbackFunction,
52 void* callbackData,
53 unsigned int callbackLevel);
54
55 OptixResult (*optixDeviceContextSetCacheEnabled)(OptixDeviceContext context, int enabled);
56
57 OptixResult (*optixDeviceContextSetCacheLocation)(OptixDeviceContext context, const char* location);
58
59 OptixResult (*optixDeviceContextSetCacheDatabaseSizes)(OptixDeviceContext context, size_t
60 lowWaterMark, size_t highWaterMark);
61
62 OptixResult (*optixDeviceContextGetCacheEnabled)(OptixDeviceContext context, int* enabled);

```

```

87
89 OptixResult (*optixDeviceContextGetCacheLocation)(OptixDeviceContext context, char* location, size_t
locationSize);
90
92 OptixResult (*optixDeviceContextGetCacheDatabaseSizes)(OptixDeviceContext context, size_t*
lowWaterMark, size_t* highWaterMark);
93
94 //@ }
96 //@ {
97
99 OptixResult (*optixModuleCreate)(OptixDeviceContext context,
100 const OptixModuleCompileOptions* moduleCompileOptions,
101 const OptixPipelineCompileOptions* pipelineCompileOptions,
102 const char* input,
103 size_t inputSize,
104 char* logString,
105 size_t* logStringSize,
106 OptixModule* module);
107
109 OptixResult (*optixModuleCreateWithTasks)(OptixDeviceContext context,
110 const OptixModuleCompileOptions* moduleCompileOptions,
111 const OptixPipelineCompileOptions* pipelineCompileOptions,
112 const char* input,
113 size_t inputSize,
114 char* logString,
115 size_t* logStringSize,
116 OptixModule* module,
117 OptixTask* firstTask);
118
120 OptixResult (*optixModuleGetCompilationState)(OptixModule module, OptixModuleCompileState* state);
121
123 OptixResult (*optixModuleDestroy)(OptixModule module);
124
126 OptixResult(*optixBuiltinISModuleGet)(OptixDeviceContext context,
127 const OptixModuleCompileOptions* moduleCompileOptions,
128 const OptixPipelineCompileOptions* pipelineCompileOptions,
129 const OptixBuiltinISOptions* builtinISOptions,
130 OptixModule* builtinModule);
131
132 //@ }
134 //@ {
135
137 OptixResult (*optixTaskExecute)(OptixTask task,
138 OptixTask* additionalTasks,
139 unsigned int maxNumAdditionalTasks,
140 unsigned int* numAdditionalTasksCreated);
141
142 //@ }
143 //@ {
144
146 OptixResult (*optixProgramGroupCreate)(OptixDeviceContext context,
147 const OptixProgramGroupDesc* programDescriptions,
148 unsigned int numProgramGroups,
149 const OptixProgramGroupOptions* options,
150 char* logString,
151 size_t* logStringSize,
152 OptixProgramGroup* programGroups);
153
155 OptixResult (*optixProgramGroupDestroy)(OptixProgramGroup programGroup);
156
158 OptixResult (*optixProgramGroupGetStackSize)(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes, OptixPipeline pipeline);
159
160 //@ }
162 //@ {
163
165 OptixResult (*optixPipelineCreate)(OptixDeviceContext context,
166 const OptixPipelineCompileOptions* pipelineCompileOptions,

```

```

167 const OptixPipelineLinkOptions* pipelineLinkOptions,
168 const OptixProgramGroup* programGroups,
169 unsigned int numProgramGroups,
170 char* logString,
171 size_t logStringSize,
172 OptixPipeline* pipeline);
173
174 OptixResult (*optixPipelineDestroy)(OptixPipeline pipeline);
175
176 OptixResult (*optixPipelineSetStackSize)(OptixPipeline pipeline,
177 unsigned int directCallableStackSizeFromTraversal,
178 unsigned int directCallableStackSizeFromState,
179 unsigned int continuationStackSize,
180 unsigned int maxTraversableGraphDepth);
181
182
183
184 //@ }
185 //@ {
186
187 OptixResult (*optixAccelComputeMemoryUsage)(OptixDeviceContext context,
188 const OptixAccelBuildOptions* accelOptions,
189 const OptixBuildInput* buildInputs,
190 unsigned int numBuildInputs,
191 OptixAccelBufferSizes* bufferSizes);
192
193
194 OptixResult (*optixAccelBuild)(OptixDeviceContext context,
195 CUstream stream,
196 const OptixAccelBuildOptions* accelOptions,
197 const OptixBuildInput* buildInputs,
198 unsigned int numBuildInputs,
199 CUdeviceptr tempBuffer,
200 size_t tempBufferSizeInBytes,
201 CUdeviceptr outputBuffer,
202 size_t outputBufferSizeInBytes,
203 OptixTraversableHandle* outputHandle,
204 const OptixAccelEmitDesc* emittedProperties,
205 unsigned int numEmittedProperties);
206
207
208 OptixResult (*optixAccelGetRelocationInfo)(OptixDeviceContext context, OptixTraversableHandle
handle, OptixRelocationInfo* info);
209
210
211
212 OptixResult (*optixCheckRelocationCompatibility)(OptixDeviceContext context,
213 const OptixRelocationInfo* info,
214 int* compatible);
215
216
217 OptixResult (*optixAccelRelocate)(OptixDeviceContext context,
218 CUstream stream,
219 const OptixRelocationInfo* info,
220 const OptixRelocateInput* relocateInputs,
221 size_t numRelocateInputs,
222 CUdeviceptr targetAccel,
223 size_t targetAccelSizeInBytes,
224 OptixTraversableHandle* targetHandle);
225
226
227
228 OptixResult (*optixAccelCompact)(OptixDeviceContext context,
229 CUstream stream,
230 OptixTraversableHandle inputHandle,
231 CUdeviceptr outputBuffer,
232 size_t outputBufferSizeInBytes,
233 OptixTraversableHandle* outputHandle);
234
235
236 OptixResult (*optixAccelEmitProperty)(OptixDeviceContext context,
237 CUstream stream,
238 OptixTraversableHandle handle,
239 const OptixAccelEmitDesc* emittedProperty);
240
241

```

```

243 OptixResult (*optixConvertPointerToTraversableHandle)(OptixDeviceContext onDevice,
244 CUdeviceptr pointer,
245 OptixTraversableType traversableType,
246 OptixTraversableHandle* traversableHandle);
247
249 OptixResult (*optixOpacityMicromapArrayComputeMemoryUsage)(OptixDeviceContext
context,
250 const OptixOpacityMicromapArrayBuildInput*
buildInput,
251 OptixMicromapBufferSizes*
bufferSizes);
252
254 OptixResult (*optixOpacityMicromapArrayBuild)(OptixDeviceContext context,
255 CUstream stream,
256 const OptixOpacityMicromapArrayBuildInput* buildInput,
257 const OptixMicromapBuffers* buffers);
258
260 OptixResult (*optixOpacityMicromapArrayGetRelocationInfo)(OptixDeviceContext context,
261 CUdeviceptr opacityMicromapArray,
262 OptixRelocationInfo* info);
263
265 OptixResult (*optixOpacityMicromapArrayRelocate)(OptixDeviceContext context,
266 CUstream stream,
267 const OptixRelocationInfo* info,
268 CUdeviceptr targetOpacityMicromapArray,
269 size_t
targetOpacityMicromapArraySizeInBytes);
270
272 OptixResult (*optixDisplacementMicromapArrayComputeMemoryUsage)(OptixDeviceContext context,
273 const
OptixDisplacementMicromapArrayBuildInput* buildInput,
274 OptixMicromapBufferSizes* bufferSizes);
275
277 OptixResult (*optixDisplacementMicromapArrayBuild)(OptixDeviceContext
context,
278 CUstream stream,
279 const OptixDisplacementMicromapArrayBuildInput*
buildInput,
280 const OptixMicromapBuffers*
buffers);
281
282 //@ }
284 //@ {
285
287 OptixResult (*optixSbtRecordPackHeader)(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer);
288
290 OptixResult (*optixLaunch)(OptixPipeline pipeline,
291 CUstream stream,
292 CUdeviceptr pipelineParams,
293 size_t pipelineParamsSize,
294 const OptixShaderBindingTable* sbt,
295 unsigned int width,
296 unsigned int height,
297 unsigned int depth);
298
299 OptixResult (*optixPlaceholder001)(OptixDeviceContext context);
300 OptixResult (*optixPlaceholder002)(OptixDeviceContext context);
301
302 //@ }
304 //@ {
305
307 OptixResult (*optixDenoiserCreate)(OptixDeviceContext context, OptixDenoiserModelKind modelKind,
const OptixDenoiserOptions* options, OptixDenoiser* returnHandle);
308
310 OptixResult (*optixDenoiserDestroy)(OptixDenoiser handle);
311

```

```

313 OptixResult (*optixDenoiserComputeMemoryResources)(const OptixDenoiser handle,
314 unsigned int maximumInputWidth,
315 unsigned int maximumInputHeight,
316 OptixDenoiserSizes* returnSizes);
317
319 OptixResult (*optixDenoiserSetup)(OptixDenoiser denoiser,
320 CUstream stream,
321 unsigned int inputWidth,
322 unsigned int inputHeight,
323 CUdeviceptr state,
324 size_t stateSizeInBytes,
325 CUdeviceptr scratch,
326 size_t scratchSizeInBytes);
327
329 OptixResult (*optixDenoiserInvoke)(OptixDenoiser denoiser,
330 CUstream stream,
331 const OptixDenoiserParams* params,
332 CUdeviceptr denoiserState,
333 size_t denoiserStateSizeInBytes,
334 const OptixDenoiserGuideLayer * guideLayer,
335 const OptixDenoiserLayer * layers,
336 unsigned int numLayers,
337 unsigned int inputOffsetX,
338 unsigned int inputOffsetY,
339 CUdeviceptr scratch,
340 size_t scratchSizeInBytes);
341
343 OptixResult (*optixDenoiserComputeIntensity)(OptixDenoiser handle,
344 CUstream stream,
345 const OptixImage2D* inputImage,
346 CUdeviceptr outputIntensity,
347 CUdeviceptr scratch,
348 size_t scratchSizeInBytes);
349
351 OptixResult (*optixDenoiserComputeAverageColor)(OptixDenoiser handle,
352 CUstream stream,
353 const OptixImage2D* inputImage,
354 CUdeviceptr outputAverageColor,
355 CUdeviceptr scratch,
356 size_t scratchSizeInBytes);
357
359 OptixResult (*optixDenoiserCreateWithUserModel)(OptixDeviceContext context, const void * data, size_t
dataSizeInBytes, OptixDenoiser* returnHandle);
360 /*@ }
361
362 } OptixFunctionTable;
363
364 // define global function table variable with ABI specific name.
365 #define OPTIX_CONCATENATE_ABI_VERSION(prefix, macro) OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro)
366 #define OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro) prefix ## _ ## macro
367 #define OPTIX_FUNCTION_TABLE_SYMBOL OPTIX_CONCATENATE_ABI_VERSION(g_optixFunctionTable,
OPTIX_ABI_VERSION)
368 // end group optix_function_table
369
370
371 #ifdef __cplusplus
372 }
373 #endif
374
375 #endif /* OPTIX_DEFINE_ABI_VERSION_ONLY */
376
377 #endif /* OPTIX_OPTIX_FUNCTION_TABLE_H */

```

## 8.15 optix\_function\_table\_definition.h File Reference

### Variables

- OptixFunctionTable OPTIX\_FUNCTION\_TABLE\_SYMBOL

### 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 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
16 * contributors may be used to endorse or promote products derived from
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
31
32
33
34
35 #ifndef OPTIX_OPTIX_FUNCTION_TABLE_DEFINITION_H
36 #define OPTIX_OPTIX_FUNCTION_TABLE_DEFINITION_H
37
38 #include "optix_function_table.h"
39
40 #ifdef __cplusplus
41 extern "C" {
42 #endif
43
44
45
46
47
48
49
50
51 OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL;
52 // end group optix_function_table
53
54
55 #ifdef __cplusplus
56 }
57 #endif
58
59 #endif // OPTIX_OPTIX_FUNCTION_TABLE_DEFINITION_H

```

## 8.17 optix\_host.h File Reference

### Macros

- #define OPTIXAPI

### Functions

- OPTIXAPI const char \* optixGetErrorName (OptixResult result)
- OPTIXAPI const char \* optixGetErrorString (OptixResult result)
- OPTIXAPI OptixResult optixDeviceContextCreate (CUcontext fromContext, const OptixDeviceContextOptions \*options, OptixDeviceContext \*context)
- OPTIXAPI OptixResult optixDeviceContextDestroy (OptixDeviceContext context)
- OPTIXAPI OptixResult optixDeviceContextGetProperty (OptixDeviceContext context, OptixDeviceProperty property, void \*value, size\_t sizeInBytes)
- OPTIXAPI OptixResult optixDeviceContextSetLogCallback (OptixDeviceContext context, OptixLogCallback callbackFunction, void \*callbackData, unsigned int callbackLevel)
- OPTIXAPI OptixResult optixDeviceContextSetCacheEnabled (OptixDeviceContext context, int enabled)
- OPTIXAPI OptixResult optixDeviceContextSetCacheLocation (OptixDeviceContext context, const char \*location)
- OPTIXAPI OptixResult optixDeviceContextSetCacheDatabaseSizes (OptixDeviceContext context, size\_t lowWaterMark, size\_t highWaterMark)
- OPTIXAPI OptixResult optixDeviceContextGetCacheEnabled (OptixDeviceContext context, int \*enabled)
- OPTIXAPI OptixResult optixDeviceContextGetCacheLocation (OptixDeviceContext context, char \*location, size\_t locationSize)
- OPTIXAPI OptixResult optixDeviceContextGetCacheDatabaseSizes (OptixDeviceContext context, size\_t \*lowWaterMark, size\_t \*highWaterMark)
- OPTIXAPI OptixResult optixPipelineCreate (OptixDeviceContext context, const OptixPipelineCompileOptions \*pipelineCompileOptions, const OptixPipelineLinkOptions \*pipelineLinkOptions, const OptixProgramGroup \*programGroups, unsigned int numProgramGroups, char \*logString, size\_t \*logStringSize, OptixPipeline \*pipeline)
- OPTIXAPI OptixResult optixPipelineDestroy (OptixPipeline pipeline)
- OPTIXAPI OptixResult optixPipelineSetStackSize (OptixPipeline pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)
- OPTIXAPI OptixResult optixModuleCreate (OptixDeviceContext context, const OptixModuleCompileOptions \*moduleCompileOptions, const OptixPipelineCompileOptions \*pipelineCompileOptions, const char \*input, size\_t inputSize, char \*logString, size\_t \*logStringSize, OptixModule \*module)
- OPTIXAPI 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)
- OPTIXAPI OptixResult optixModuleGetCompilationState (OptixModule module, OptixModuleCompileState \*state)
- OPTIXAPI OptixResult optixModuleDestroy (OptixModule module)
- OPTIXAPI OptixResult optixBuiltinISModuleGet (OptixDeviceContext context, const OptixModuleCompileOptions \*moduleCompileOptions, const OptixPipelineCompileOptions \*pipelineCompileOptions, const OptixBuiltinISOOptions \*builtinISOOptions, OptixModule \*builtinModule)
- OPTIXAPI OptixResult optixTaskExecute (OptixTask task, OptixTask \*additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int \*numAdditionalTasksCreated)

- OPTIXAPI `OptixResult optixProgramGroupGetStackSize` (`OptixProgramGroup` programGroup, `OptixStackSizes` \*stackSizes, `OptixPipeline` pipeline)
- OPTIXAPI `OptixResult optixProgramGroupCreate` (`OptixDeviceContext` context, const `OptixProgramGroupDesc` \*programDescriptions, unsigned int numProgramGroups, const `OptixProgramGroupOptions` \*options, char \*logString, size\_t \*logStringSize, `OptixProgramGroup` \*programGroups)
- OPTIXAPI `OptixResult optixProgramGroupDestroy` (`OptixProgramGroup` programGroup)
- OPTIXAPI `OptixResult optixLaunch` (`OptixPipeline` pipeline, `CUstream` stream, `CUdeviceptr` pipelineParams, size\_t pipelineParamsSize, const `OptixShaderBindingTable` \*sbt, unsigned int width, unsigned int height, unsigned int depth)
- OPTIXAPI `OptixResult optixSbtRecordPackHeader` (`OptixProgramGroup` programGroup, void \*sbtRecordHeaderHostPointer)
- OPTIXAPI `OptixResult optixAccelComputeMemoryUsage` (`OptixDeviceContext` context, const `OptixAccelBuildOptions` \*accelOptions, const `OptixBuildInput` \*buildInputs, unsigned int numBuildInputs, `OptixAccelBufferSizes` \*bufferSizes)
- OPTIXAPI `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)
- OPTIXAPI `OptixResult optixAccelGetRelocationInfo` (`OptixDeviceContext` context, `OptixTraversableHandle` handle, `OptixRelocationInfo` \*info)
- OPTIXAPI `OptixResult optixCheckRelocationCompatibility` (`OptixDeviceContext` context, const `OptixRelocationInfo` \*info, int \*compatible)
- OPTIXAPI `OptixResult optixAccelRelocate` (`OptixDeviceContext` context, `CUstream` stream, const `OptixRelocationInfo` \*info, const `OptixRelocateInput` \*relocateInputs, size\_t numRelocateInputs, `CUdeviceptr` targetAccel, size\_t targetAccelSizeInBytes, `OptixTraversableHandle` \*targetHandle)
- OPTIXAPI `OptixResult optixAccelCompact` (`OptixDeviceContext` context, `CUstream` stream, `OptixTraversableHandle` inputHandle, `CUdeviceptr` outputBuffer, size\_t outputBufferSizeInBytes, `OptixTraversableHandle` \*outputHandle)
- OPTIXAPI `OptixResult optixAccelEmitProperty` (`OptixDeviceContext` context, `CUstream` stream, `OptixTraversableHandle` handle, const `OptixAccelEmitDesc` \*emittedProperty)
- OPTIXAPI `OptixResult optixConvertPointerToTraversableHandle` (`OptixDeviceContext` onDevice, `CUdeviceptr` pointer, `OptixTraversableType` traversableType, `OptixTraversableHandle` \*traversableHandle)
- OPTIXAPI `OptixResult optixOpacityMicromapArrayComputeMemoryUsage` (`OptixDeviceContext` context, const `OptixOpacityMicromapArrayBuildInput` \*buildInput, `OptixMicromapBufferSizes` \*bufferSizes)
- OPTIXAPI `OptixResult optixOpacityMicromapArrayBuild` (`OptixDeviceContext` context, `CUstream` stream, const `OptixOpacityMicromapArrayBuildInput` \*buildInput, const `OptixMicromapBuffers` \*buffers)
- OPTIXAPI `OptixResult optixOpacityMicromapArrayGetRelocationInfo` (`OptixDeviceContext` context, `CUdeviceptr` opacityMicromapArray, `OptixRelocationInfo` \*info)
- OPTIXAPI `OptixResult optixOpacityMicromapArrayRelocate` (`OptixDeviceContext` context, `CUstream` stream, const `OptixRelocationInfo` \*info, `CUdeviceptr` targetOpacityMicromapArray, size\_t targetOpacityMicromapArraySizeInBytes)
- OPTIXAPI `OptixResult optixDisplacementMicromapArrayComputeMemoryUsage` (`OptixDeviceContext` context, const `OptixDisplacementMicromapArrayBuildInput` \*buildInput, `OptixMicromapBufferSizes` \*bufferSizes)
- OPTIXAPI `OptixResult optixDisplacementMicromapArrayBuild` (`OptixDeviceContext` context, `CUstream` stream, const `OptixDisplacementMicromapArrayBuildInput` \*buildInput, const `OptixMicromapBuffers` \*buffers)

- OPTIXAPI `OptixResult optixDenoiserCreate` (`OptixDeviceContext context`, `OptixDenoiserModelKind modelKind`, `const OptixDenoiserOptions *options`, `OptixDenoiser *denoiser`)
- OPTIXAPI `OptixResult optixDenoiserCreateWithUserModel` (`OptixDeviceContext context`, `const void *userData`, `size_t userDataSizeInBytes`, `OptixDenoiser *denoiser`)
- OPTIXAPI `OptixResult optixDenoiserDestroy` (`OptixDenoiser denoiser`)
- OPTIXAPI `OptixResult optixDenoiserComputeMemoryResources` (`const OptixDenoiser denoiser`, `unsigned int outputWidth`, `unsigned int outputHeight`, `OptixDenoiserSizes *returnSizes`)
- OPTIXAPI `OptixResult optixDenoiserSetup` (`OptixDenoiser denoiser`, `CUstream stream`, `unsigned int inputWidth`, `unsigned int inputHeight`, `CUdeviceptr denoiserState`, `size_t denoiserStateSizeInBytes`, `CUdeviceptr scratch`, `size_t scratchSizeInBytes`)
- OPTIXAPI `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`)
- OPTIXAPI `OptixResult optixDenoiserComputeIntensity` (`OptixDenoiser denoiser`, `CUstream stream`, `const OptixImage2D *inputImage`, `CUdeviceptr outputIntensity`, `CUdeviceptr scratch`, `size_t scratchSizeInBytes`)
- OPTIXAPI `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 Macro Definition Documentation

#### 8.17.2.1 OPTIXAPI

```
#define OPTIXAPI
```

Mixing multiple SDKs in a single application will result in symbol collisions. To enable different compilation units to use different SDKs, use `OPTIX_ENABLE_SDK_MIXING`.

### 8.17.3 Function Documentation

#### 8.17.3.1 optixAccelBuild()

```
OPTIXAPI 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>                 |                                                                      |
| in  | <i>stream</i>                  |                                                                      |
| in  | <i>accelOptions</i>            | accel options                                                        |
| in  | <i>buildInputs</i>             | an array of <code>OptixBuildInput</code> objects                     |
| in  | <i>numBuildInputs</i>          | must be $\geq 1$ for GAS, and $= 1$ for IAS                          |
| in  | <i>tempBuffer</i>              | must be a multiple of <code>OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT</code> |
| in  | <i>tempBufferSizeInBytes</i>   |                                                                      |
| in  | <i>outputBuffer</i>            | must be a multiple of <code>OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT</code> |
| 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)            |

#### 8.17.3.2 optixAccelCompact()

```

OPTIXAPI 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>  |

## Parameters

|     |                                |
|-----|--------------------------------|
| in  | <i>inputHandle</i>             |
| in  | <i>outputBuffer</i>            |
| in  | <i>outputBufferSizeInBytes</i> |
| out | <i>outputHandle</i>            |

## 8.17.3.3 optixAccelComputeMemoryUsage()

```
OPTIXAPI 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                                  |

## 8.17.3.4 optixAccelEmitProperty()

```
OPTIXAPI 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 |

## 8.17.3.5 optixAccelGetRelocationInfo()

```
OPTIXAPI 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.

### 8.17.3.6 optixAccelRelocate()

```

OPTIXAPI 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>                |
| 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>           |

#### 8.17.3.7 optixBuiltinISModuleGet()

```
OPTIXAPI 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.)

#### 8.17.3.8 optixCheckRelocationCompatibility()

```
OPTIXAPI 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>    |  |

## Parameters

|            |                   |                                                                                                                                                                                                                                                                |
|------------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>out</b> | <i>compatible</i> | If OPTIX_SUCCESS is returned 'compatible' will have the value of either: <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> |
|------------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 8.17.3.9 optixConvertPointerToTraversableHandle()

```

OPTIXAPI OptixResult optixConvertPointerToTraversableHandle (
 OptixDeviceContext onDevice,
 CUdeviceptr pointer,
 OptixTraversableType traversableType,
 OptixTraversableHandle * traversableHandle)

```

## Parameters

|            |                          |                                                                                                                           |
|------------|--------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <b>in</b>  | <i>onDevice</i>          |                                                                                                                           |
| <b>in</b>  | <i>pointer</i>           | pointer to traversable allocated in OptixDeviceContext. This pointer must be a multiple of OPTIX_TRANSFORM_BYTE_ALIGNMENT |
| <b>in</b>  | <i>traversableType</i>   | Type of OptixTraversableHandle to create                                                                                  |
| <b>out</b> | <i>traversableHandle</i> | traversable handle. traversableHandle must be in host memory                                                              |

## 8.17.3.10 optixDenoiserComputeAverageColor()

```

OPTIXAPI 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

|            |                           |              |
|------------|---------------------------|--------------|
| <b>in</b>  | <i>denoiser</i>           |              |
| <b>in</b>  | <i>stream</i>             |              |
| <b>in</b>  | <i>inputImage</i>         |              |
| <b>out</b> | <i>outputAverageColor</i> | three floats |
| <b>in</b>  | <i>scratch</i>            |              |
| <b>in</b>  | <i>scratchSizeInBytes</i> |              |

### 8.17.3.11 optixDenoiserComputeIntensity()

```
OPTIXAPI 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> |              |

### 8.17.3.12 optixDenoiserComputeMemoryResources()

```
OPTIXAPI 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> |
| out | <i>returnSizes</i>  |

### 8.17.3.13 optixDenoiserCreate()

```
OPTIXAPI 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>  |

### 8.17.3.14 optixDenoiserCreateWithUserModel()

```
OPTIXAPI 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> |
|----|----------------|

## Parameters

|     |                            |
|-----|----------------------------|
| in  | <i>userData</i>            |
| in  | <i>userDataSizeInBytes</i> |
| out | <i>denoiser</i>            |

## 8.17.3.15 optixDenoiserDestroy()

```
OPTIXAPI OptixResult optixDenoiserDestroy (
 OptixDenoiser denoiser)
```

Destroys the denoiser object and any associated host resources.

## 8.17.3.16 optixDenoiserInvoke()

```
OPTIXAPI 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>       |

#### 8.17.3.17 optixDenoiserSetup()

```
OPTIXAPI 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>       |

### 8.17.3.18 optixDeviceContextCreate()

```

OPTIXAPI 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

### 8.17.3.19 optixDeviceContextDestroy()

```
OPTIXAPI 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.

### 8.17.3.20 optixDeviceContextGetCacheDatabaseSizes()

```
OPTIXAPI 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 |

### 8.17.3.21 optixDeviceContextGetCacheEnabled()

```
OPTIXAPI 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 |

### 8.17.3.22 optixDeviceContextGetCacheLocation()

```
OPTIXAPI 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                                                 |

## 8.17.3.23 optixDeviceContextGetProperty()

```
OPTIXAPI 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                               |

## 8.17.3.24 optixDeviceContextSetCacheDatabaseSizes()

```
OPTIXAPI 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 |

8.17.3.25 `optixDeviceContextSetCacheEnabled()`

```
OPTIXAPI 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 |

8.17.3.26 `optixDeviceContextSetCacheLocation()`

```
OPTIXAPI 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 |

## 8.17.3.27 optixDeviceContextSetLogCallback()

```
OPTIXAPI 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                                                |

## 8.17.3.28 optixDisplacementMicromapArrayBuild()

```
OPTIXAPI OptixResult optixDisplacementMicromapArrayBuild (
 OptixDeviceContext context,
 CUstream stream,
 const OptixDisplacementMicromapArrayBuildInput * buildInput,
 const OptixMicromapBuffers * buffers)
```

Construct an array of Displacement Micromaps (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> |  |
|----|----------------|--|

## Parameters

|    |                   |                                                   |
|----|-------------------|---------------------------------------------------|
| 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                        |

## 8.17.3.29 optixDisplacementMicromapArrayComputeMemoryUsage()

```
OPTIXAPI 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> |  |

## 8.17.3.30 optixGetErrorName()

```
OPTIXAPI 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](#)

## 8.17.3.31 optixGetErrorString()

```
OPTIXAPI 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](#)

### 8.17.3.32 optixLaunch()

```
OPTIXAPI 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.

### 8.17.3.33 optixModuleCreate()

```
OPTIXAPI 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.

#### 8.17.3.34 optixModuleCreateWithTasks()

```

OPTIXAPI 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`

### 8.17.3.35 optixModuleDestroy()

```
OPTIXAPI 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.

### 8.17.3.36 optixModuleGetCompilationState()

```
OPTIXAPI 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`

### 8.17.3.37 optixOpacityMicromapArrayBuild()

```
OPTIXAPI 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                                     |

### 8.17.3.38 optixOpacityMicromapArrayComputeMemoryUsage()

```
OPTIXAPI 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> |  |

### 8.17.3.39 optixOpacityMicromapArrayGetRelocationInfo()

```
OPTIXAPI 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>                 |

#### 8.17.3.40 optixOpacityMicromapArrayRelocate()

```

OPTIXAPI 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> |

## 8.17.3.41 optixPipelineCreate()

```

OPTIXAPI 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 <i>logStringSize</i> > 0 <i>logString</i> will be null terminated. |
| in, out | <i>logStringSize</i>          |                                                                                                                   |
| out     | <i>pipeline</i>               |                                                                                                                   |

## 8.17.3.42 optixPipelineDestroy()

```

OPTIXAPI 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.

### 8.17.3.43 optixPipelineSetStackSize()

```
OPTIXAPI 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 programs is at most 2, and no DC programs or 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 | <code>pipeline</code>                             | The pipeline to configure the stack size for.                                      |
| in | <code>directCallableStackSizeFromTraversal</code> | The direct stack size requirement for direct callables invoked from IS or AH.      |
| in | <code>directCallableStackSizeFromState</code>     | The direct stack size requirement for direct callables invoked from RG, MS, or CH. |
| in | <code>continuationStackSize</code>                | The continuation stack requirement.                                                |
| in | <code>maxTraversableGraphDepth</code>             | The maximum depth of a traversable graph passed to trace.                          |

### 8.17.3.44 optixProgramGroupCreate()

```
OPTIXAPI 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      | <code>context</code>             |                                                                                                                                  |
| in      | <code>programDescriptions</code> | <code>N * OptixProgramGroupDesc</code>                                                                                           |
| in      | <code>numProgramGroups</code>    | <code>N</code>                                                                                                                   |
| in      | <code>options</code>             |                                                                                                                                  |
| out     | <code>logString</code>           | Information will be written to this string. If <code>logStringSize &gt; 0</code> <code>logString</code> will be null terminated. |
| in, out | <code>logStringSize</code>       |                                                                                                                                  |
| out     | <code>programGroups</code>       |                                                                                                                                  |

#### 8.17.3.45 optixProgramGroupDestroy()

`OPTIXAPI 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.

#### 8.17.3.46 optixProgramGroupGetStackSize()

`OPTIXAPI 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  | <code>programGroup</code> | the program group             |
| out | <code>stackSizes</code>   | the corresponding stack sizes |

## Parameters

|    |                 |                                                                      |
|----|-----------------|----------------------------------------------------------------------|
| in | <i>pipeline</i> | considering the program group within the given pipeline, can be NULL |
|----|-----------------|----------------------------------------------------------------------|

## 8.17.3.47 optixSbtRecordPackHeader()

```
OPTIXAPI 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                |

## 8.17.3.48 optixTaskExecute()

```
OPTIXAPI 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 <code>OptixTask</code> to execute                                                                   |
| in  | <i>additionalTasks</i>           | pointer to array of <code>OptixTask</code> objects to be filled in                                      |
| in  | <i>maxNumAdditionalTasks</i>     | maximum number of additional <code>OptixTask</code> objects                                             |
| out | <i>numAdditionalTasksCreated</i> | number of <code>OptixTask</code> objects created by OptiX and written into <code>additionalTasks</code> |

## 8.18 optix\_host.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2010 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
4 *
5 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
6 * property and proprietary rights in and to this material, related
```

```

7 * documentation and any modifications thereto. Any use, reproduction,
8 * disclosure or distribution of this material and related documentation
9 * without an express license agreement from NVIDIA CORPORATION or
10 * its affiliates is strictly prohibited.
11 */
18
19 #ifndef OPTIX_OPTIX_HOST_H
20 #define OPTIX_OPTIX_HOST_H
21
24 #ifndef OPTIXAPI
25 # ifdef OPTIX_ENABLE_SDK_MIXING
26 # define OPTIXAPI static
27 # else // OPTIX_ENABLE_SDK_MIXING
28 # ifdef __cplusplus
29 # define OPTIXAPI extern "C"
30 # else // __cplusplus
31 # define OPTIXAPI
32 # endif // __cplusplus
33 # endif // OPTIX_ENABLE_SDK_MIXING
34 #endif // OPTIXAPI
35
36 #include "optix_types.h"
37 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
38 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver types must be defined through other
39 // means before including optix headers.
40 #include <cuda.h>
41 #endif
42
43 #ifdef NV_MODULE_OPTIX
44 // This is a mechanism to include <g_nvconfig.h> in driver builds only and translate any nvconfig macro to
45 // a custom OPTIX-specific macro, that can also be used in SDK builds/installs
46 #include <exp/misc/optix_nvconfig_translate.h> // includes <g_nvconfig.h>
47 #endif // NV_MODULE_OPTIX
48
49
50
51
52
53
54
55
56 OPTIXAPI const char* optixGetErrorName(OptixResult result);
57
58
59 OPTIXAPI const char* optixGetErrorString(OptixResult result);
60
61
62
63
64
65
66
67
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
103 OPTIXAPI OptixResult optixDeviceContextCreate(CUcontext fromContext, const OptixDeviceContextOptions*
options, OptixDeviceContext* context);
104
105
106
107
108
109
110
111
112
113 OPTIXAPI OptixResult optixDeviceContextDestroy(OptixDeviceContext context);
114
115
116
117
118
119
120
121 OPTIXAPI OptixResult optixDeviceContextGetProperty(OptixDeviceContext context, OptixDeviceProperty
property, void* value, size_t sizeInBytes);
122
123
124
125
126 OPTIXAPI OptixResult optixDeviceContextSetLogCallback(OptixDeviceContext context,
127 OptixLogCallback callbackFunction,
128 void* callbackData,
129 unsigned int callbackLevel);
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149 OPTIXAPI OptixResult optixDeviceContextSetCacheEnabled(OptixDeviceContext context, int enabled);
150
151
152
153
154
155
156
157
158
159
160
161 OPTIXAPI OptixResult optixDeviceContextSetCacheLocation(OptixDeviceContext context, const char*
location);
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 OPTIXAPI OptixResult optixDeviceContextGetCacheEnabled(OptixDeviceContext context, int* enabled);

```

```

223 OPTIXAPI OptixResult optixDeviceContextGetCacheLocation(OptixDeviceContext context, char* location,
size_t locationSize);
224
232 OPTIXAPI OptixResult optixDeviceContextGetCacheDatabaseSizes(OptixDeviceContext context, size_t*
lowWaterMark, size_t* highWaterMark);
233
235
237
238
262 OPTIXAPI 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 OPTIXAPI OptixResult optixPipelineDestroy(OptixPipeline pipeline);
273
296 OPTIXAPI 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 OPTIXAPI 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
385 OPTIXAPI 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 OPTIXAPI OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState* state);
402
408 OPTIXAPI OptixResult optixModuleDestroy(OptixModule module);
409
413 OPTIXAPI OptixResult optixBuiltinISModuleGet(OptixDeviceContext context,
414 const OptixModuleCompileOptions* moduleCompileOptions,
415 const OptixPipelineCompileOptions* pipelineCompileOptions,
416 const OptixBuiltinISOptions* builtinISOptions,
417 OptixModule* builtinModule);
418
420
422
423
441 OPTIXAPI OptixResult optixTaskExecute(OptixTask task,
442 OptixTask* additionalTasks,
443 unsigned int maxNumAdditionalTasks,
444 unsigned int* numAdditionalTasksCreated);
445
447

```

```

449
450
459 OPTIXAPI OptixResult optixProgramGroupGetStackSize(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes, OptixPipeline pipeline);
460
486 OPTIXAPI OptixResult optixProgramGroupCreate(OptixDeviceContext context,
487 const OptixProgramGroupDesc* programDescriptions,
488 unsigned int numProgramGroups,
489 const OptixProgramGroupOptions* options,
490 char* logString,
491 size_t* logStringSize,
492 OptixProgramGroup* programGroups);
493
495 OPTIXAPI OptixResult optixProgramGroupDestroy(OptixProgramGroup programGroup);
496
498
500
501
528 OPTIXAPI OptixResult optixLaunch(OptixPipeline pipeline,
529 CUstream stream,
530 CUdeviceptr pipelineParams,
531 size_t pipelineParamsSize,
532 const OptixShaderBindingTable* sbt,
533 unsigned int width,
534 unsigned int height,
535 unsigned int depth);
536
539 OPTIXAPI OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer);
540
542
544
545
551 OPTIXAPI OptixResult optixAccelComputeMemoryUsage(OptixDeviceContext context,
552 const OptixAccelBuildOptions* accelOptions,
553 const OptixBuildInput* buildInputs,
554 unsigned int numBuildInputs,
555 OptixAccelBufferSizes* bufferSizes);
556
569 OPTIXAPI OptixResult optixAccelBuild(OptixDeviceContext context,
570 CUstream stream,
571 const OptixAccelBuildOptions* accelOptions,
572 const OptixBuildInput* buildInputs,
573 unsigned int numBuildInputs,
574 CUdeviceptr tempBuffer,
575 size_t tempBufferSizeInBytes,
576 CUdeviceptr outputBuffer,
577 size_t outputBufferSizeInBytes,
578 OptixTraversableHandle* outputHandle,
579 const OptixAccelEmitDesc* emittedProperties,
580 unsigned int numEmittedProperties);
581
599 OPTIXAPI OptixResult optixAccelGetRelocationInfo(OptixDeviceContext context, OptixTraversableHandle
handle, OptixRelocationInfo* info);
600
612 OPTIXAPI OptixResult optixCheckRelocationCompatibility(OptixDeviceContext context, const
OptixRelocationInfo* info, int* compatible);
613
651 OPTIXAPI OptixResult optixAccelRelocate(OptixDeviceContext context,
652 CUstream stream,
653 const OptixRelocationInfo* info,
654 const OptixRelocateInput* relocateInputs,
655 size_t numRelocateInputs,
656 CUdeviceptr targetAccel,
657 size_t targetAccelSizeInBytes,
658 OptixTraversableHandle* targetHandle);
659

```

```

677 OPTIXAPI OptixResult optixAccelCompact(OptixDeviceContext context,
678 CUstream stream,
679 OptixTraversableHandle inputHandle,
680 CUdeviceptr outputBuffer,
681 size_t outputBufferSizeInBytes,
682 OptixTraversableHandle* outputHandle);
683
692 OPTIXAPI OptixResult optixAccelEmitProperty(OptixDeviceContext context,
693 CUstream stream,
694 OptixTraversableHandle handle,
695 const OptixAccelEmitDesc* emittedProperty);
696
701 OPTIXAPI OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext onDevice,
702 CUdeviceptr pointer,
703 OptixTraversableType traversableType,
704 OptixTraversableHandle* traversableHandle);
705
706
712 OPTIXAPI OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext
context,
713 const OptixOpacityMicromapArrayBuildInput*
buildInput,
714 OptixMicromapBufferSizes* bufferSizes);
715
738 OPTIXAPI OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext context,
739 CUstream stream,
740 const OptixOpacityMicromapArrayBuildInput* buildInput,
741 const OptixMicromapBuffers* buffers);
742
758 OPTIXAPI OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext context,
759 CUdeviceptr opacityMicromapArray,
760 OptixRelocationInfo* info);
761
788 OPTIXAPI OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext context,
789 CUstream stream,
790 const OptixRelocationInfo* info,
791 CUdeviceptr targetOpacityMicromapArray,
792 size_t targetOpacityMicromapArraySizeInBytes);
793
799 OPTIXAPI OptixResult optixDisplacementMicromapArrayComputeMemoryUsage(OptixDeviceContext context,
800 const
OptixDisplacementMicromapArrayBuildInput* buildInput,
801 OptixMicromapBufferSizes* bufferSizes);
802
815 OPTIXAPI OptixResult optixDisplacementMicromapArrayBuild(OptixDeviceContext
context,
816 CUstream stream,
817 const OptixDisplacementMicromapArrayBuildInput*
buildInput,
818 const OptixMicromapBuffers*
buffers);
819
820
822
824
825
837 OPTIXAPI OptixResult optixDenoiserCreate(OptixDeviceContext context,
838 OptixDenoiserModelKind modelKind,
839 const OptixDenoiserOptions* options,
840 OptixDenoiser* denoiser);
841
854 OPTIXAPI OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context,
855 const void* userData,
856 size_t userDataSizeInBytes,
857 OptixDenoiser* denoiser);
858
860 OPTIXAPI OptixResult optixDenoiserDestroy(OptixDenoiser denoiser);

```

```

861
881 OPTIXAPI OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser denoiser,
882 unsigned int outputWidth,
883 unsigned int outputHeight,
884 OptixDenoiserSizes* returnSizes);
885
902 OPTIXAPI OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
903 CUstream stream,
904 unsigned int inputWidth,
905 unsigned int inputHeight,
906 CUdeviceptr denoiserState,
907 size_t denoiserStateSizeInBytes,
908 CUdeviceptr scratch,
909 size_t scratchSizeInBytes);
910
976 OPTIXAPI OptixResult optixDenoiserInvoke(OptixDenoiser denoiser,
977 CUstream stream,
978 const OptixDenoiserParams* params,
979 CUdeviceptr denoiserState,
980 size_t denoiserStateSizeInBytes,
981 const OptixDenoiserGuideLayer* guideLayer,
982 const OptixDenoiserLayer* layers,
983 unsigned int numLayers,
984 unsigned int inputOffsetX,
985 unsigned int inputOffsetY,
986 CUdeviceptr scratch,
987 size_t scratchSizeInBytes);
988
1012 OPTIXAPI OptixResult optixDenoiserComputeIntensity(OptixDenoiser denoiser,
1013 CUstream stream,
1014 const OptixImage2D* inputImage,
1015 CUdeviceptr outputIntensity,
1016 CUdeviceptr scratch,
1017 size_t scratchSizeInBytes);
1018
1033 OPTIXAPI OptixResult optixDenoiserComputeAverageColor(OptixDenoiser denoiser,
1034 CUstream stream,
1035 const OptixImage2D* inputImage,
1036 CUdeviceptr outputAverageColor,
1037 CUdeviceptr scratch,
1038 size_t scratchSizeInBytes);
1039
1041
1042 #include "optix_function_table.h"
1043
1044 #endif // OPTIX_OPTIX_HOST_H

```

## 8.19 optix\_micromap.h File Reference

### Functions

- `OPTIX_MICROMAP_INLINE_FUNC` void `optixMicromapIndexToBaseBarycentrics` (unsigned int micromapTriangleIndex, unsigned int subdivisionLevel, float2 &baseBarycentrics0, float2 &baseBarycentrics1, float2 &baseBarycentrics2)
- `OPTIX_MICROMAP_INLINE_FUNC` float2 `optixBaseBarycentricsToMicroBarycentrics` (float2 baseBarycentrics, float2 microVertexBaseBarycentrics[3])

### 8.19.1 Detailed Description

OptiX micromap helper functions.

Author

NVIDIA Corporation

OptiX micromap helper functions. Useable on either host or device.

## 8.19.2 Function Documentation

### 8.19.2.1 optixBaseBarycentricsToMicroBarycentrics()

```
OPTIX_MICROMAP_INLINE_FUNC float2 optixBaseBarycentricsToMicroBarycentrics (
 float2 baseBarycentrics,
 float2 microVertexBaseBarycentrics[3])
```

Maps barycentrics in the space of the base triangle to barycentrics of a micro triangle. The vertices of the micro triangle are defined by its barycentrics in the space of the base triangle. These can be queried for a DMM hit by using `optixGetMicroTriangleBarycentricsData()`.

### 8.19.2.2 optixMicromapIndexToBaseBarycentrics()

```
OPTIX_MICROMAP_INLINE_FUNC void optixMicromapIndexToBaseBarycentrics (
 unsigned int micromapTriangleIndex,
 unsigned int subdivisionLevel,
 float2 & baseBarycentrics0,
 float2 & baseBarycentrics1,
 float2 & baseBarycentrics2)
```

Converts a micromap triangle index to the three base-triangle barycentric coordinates of the micro-triangle vertices in the base triangle. The base triangle is the triangle that the micromap is applied to. Note that for displaced micro-meshes this function can be used to compute a UV mapping from sub triangle to base triangle.

Parameters

|     |                              |                                                                                                                    |
|-----|------------------------------|--------------------------------------------------------------------------------------------------------------------|
| in  | <i>micromapTriangleIndex</i> | Index of a micro- or sub triangle within a micromap.                                                               |
| in  | <i>subdivisionLevel</i>      | Number of subdivision levels of the micromap or number of subdivision levels being considered (for sub triangles). |
| out | <i>baseBarycentrics0</i>     | Barycentric coordinates in the space of the base triangle of vertex 0 of the micromap triangle.                    |
| out | <i>baseBarycentrics1</i>     | Barycentric coordinates in the space of the base triangle of vertex 1 of the micromap triangle.                    |
| out | <i>baseBarycentrics2</i>     | Barycentric coordinates in the space of the base triangle of vertex 2 of the micromap triangle.                    |

## 8.20 optix\_micromap.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2022 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
```

```

7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
16 * contributors may be used to endorse or promote products derived from
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
31 #ifndef OPTIX_OPTIX_MICROMAP_H
32 #define OPTIX_OPTIX_MICROMAP_H
33
34 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
35 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver type float2 must be defined through other
36 // means before including optix headers.
37 #include <vector_types.h>
38 #endif
39 #include "internal/optix_micromap_impl.h"
40
41 OPTIX_MICROMAP_INLINE_FUNC void optixMicromapIndexToBaseBarycentrics(unsigned int micromapTriangleIndex,
42 unsigned int subdivisionLevel,
43 float2& baseBarycentrics0,
44 float2& baseBarycentrics1,
45 float2& baseBarycentrics2)
46 {
47 optix_impl::micro2bary(micromapTriangleIndex, subdivisionLevel, baseBarycentrics0, baseBarycentrics1,
48 baseBarycentrics2);
49 }
50
51 OPTIX_MICROMAP_INLINE_FUNC float2 optixBaseBarycentricsToMicroBarycentrics(float2 baseBarycentrics,
52 float2 microVertexBaseBarycentrics[3])
53 {
54 return optix_impl::base2micro(baseBarycentrics, microVertexBaseBarycentrics);
55 }
56 #endif // OPTIX_OPTIX_MICROMAP_H

```

## 8.21 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)

### 8.21.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

## 8.22 optix\_stack\_size.h

[Go to the documentation of this file.](#)

```

1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
16 * contributors may be used to endorse or promote products derived from
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
31
32
33
34
35 #ifndef OPTIX_OPTIX_STACK_SIZE_H
36 #define OPTIX_OPTIX_STACK_SIZE_H
37
38 #include "optix.h"
39

```

```

40 #include <algorithm>
41 #include <cstring>
42
43 #ifdef __cplusplus
44 extern "C" {
45 #endif
46
47 inline OptixResult optixUtilAccumulateStackSizes(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes, OptixPipeline pipeline)
48 {
49 if(!stackSizes)
50 return OPTIX_ERROR_INVALID_VALUE;
51
52 OptixStackSizes localStackSizes;
53 OptixResult result = optixProgramGroupGetStackSize(programGroup, &localStackSizes, pipeline);
54 if(result != OPTIX_SUCCESS)
55 return result;
56
57 stackSizes->cssRG = std::max(stackSizes->cssRG, localStackSizes.cssRG);
58 stackSizes->cssMS = std::max(stackSizes->cssMS, localStackSizes.cssMS);
59 stackSizes->cssCH = std::max(stackSizes->cssCH, localStackSizes.cssCH);
60 stackSizes->cssAH = std::max(stackSizes->cssAH, localStackSizes.cssAH);
61 stackSizes->cssIS = std::max(stackSizes->cssIS, localStackSizes.cssIS);
62 stackSizes->cssCC = std::max(stackSizes->cssCC, localStackSizes.cssCC);
63 stackSizes->dssDC = std::max(stackSizes->dssDC, localStackSizes.dssDC);
64
65 return OPTIX_SUCCESS;
66 }
67
68 inline OptixResult optixUtilComputeStackSizes(const OptixStackSizes* stackSizes,
69 unsigned int maxTraceDepth,
70 unsigned int maxCCDepth,
71 unsigned int maxDCDepth,
72 unsigned int* directCallableStackSizeFromTraversal,
73 unsigned int* directCallableStackSizeFromState,
74 unsigned int* continuationStackSize)
75 {
76 if(!stackSizes)
77 return OPTIX_ERROR_INVALID_VALUE;
78
79 const unsigned int cssRG = stackSizes->cssRG;
80 const unsigned int cssMS = stackSizes->cssMS;
81 const unsigned int cssCH = stackSizes->cssCH;
82 const unsigned int cssAH = stackSizes->cssAH;
83 const unsigned int cssIS = stackSizes->cssIS;
84 const unsigned int cssCC = stackSizes->cssCC;
85 const unsigned int dssDC = stackSizes->dssDC;
86
87 if(directCallableStackSizeFromTraversal)
88 *directCallableStackSizeFromTraversal = maxDCDepth * dssDC;
89 if(directCallableStackSizeFromState)
90 *directCallableStackSizeFromState = maxDCDepth * dssDC;
91
92 // upper bound on continuation stack used by call trees of continuation callables
93 unsigned int cssCCTree = maxCCDepth * cssCC;
94
95 // upper bound on continuation stack used by CH or MS programs including the call tree of
96 // continuation callables
97 unsigned int cssCHorMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
98
99 // clang-format off
100 if(continuationStackSize)
101 *continuationStackSize
102 = cssRG + cssCCTree
103 + (std::max(maxTraceDepth, 1u) - 1) * cssCHorMSPlusCCTree
104 + std::min(maxTraceDepth, 1u) * std::max(cssCHorMSPlusCCTree, cssIS + cssAH);
105 // clang-format on

```

```

128
129 return OPTIX_SUCCESS;
130 }
131
155 inline OptixResult optixUtilComputeStackSizesDCSplit(const OptixStackSizes* stackSizes,
156 unsigned int dssDCFromTraversal,
157 unsigned int dssDCFromState,
158 unsigned int maxTraceDepth,
159 unsigned int maxCCDepth,
160 unsigned int maxDCDepthFromTraversal,
161 unsigned int maxDCDepthFromState,
162 unsigned int*
directCallableStackSizeFromTraversal,
163 unsigned int* directCallableStackSizeFromState,
164 unsigned int* continuationStackSize)
165 {
166 if(!stackSizes)
167 return OPTIX_ERROR_INVALID_VALUE;
168
169 const unsigned int cssRG = stackSizes->cssRG;
170 const unsigned int cssMS = stackSizes->cssMS;
171 const unsigned int cssCH = stackSizes->cssCH;
172 const unsigned int cssAH = stackSizes->cssAH;
173 const unsigned int cssIS = stackSizes->cssIS;
174 const unsigned int cssCC = stackSizes->cssCC;
175 // use dssDCFromTraversal and dssDCFromState instead of stackSizes->dssDC
176
177 if(directCallableStackSizeFromTraversal)
178 *directCallableStackSizeFromTraversal = maxDCDepthFromTraversal * dssDCFromTraversal;
179 if(directCallableStackSizeFromState)
180 *directCallableStackSizeFromState = maxDCDepthFromState * dssDCFromState;
181
182 // upper bound on continuation stack used by call trees of continuation callables
183 unsigned int cssCCTree = maxCCDepth * cssCC;
184
185 // upper bound on continuation stack used by CH or MS programs including the call tree of
186 // continuation callables
187 unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
188
189 // clang-format off
190 if(continuationStackSize)
191 *continuationStackSize
192 = cssRG + cssCCTree
193 + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
194 + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
195 // clang-format on
196
197 return OPTIX_SUCCESS;
198 }
199
216 inline OptixResult optixUtilComputeStackSizesCssCCTree(const OptixStackSizes* stackSizes,
217 unsigned int cssCCTree,
218 unsigned int maxTraceDepth,
219 unsigned int maxDCDepth,
220 unsigned int*
directCallableStackSizeFromTraversal,
221 unsigned int* directCallableStackSizeFromState,
222 unsigned int* continuationStackSize)
223 {
224 if(!stackSizes)
225 return OPTIX_ERROR_INVALID_VALUE;
226
227 const unsigned int cssRG = stackSizes->cssRG;
228 const unsigned int cssMS = stackSizes->cssMS;
229 const unsigned int cssCH = stackSizes->cssCH;
230 const unsigned int cssAH = stackSizes->cssAH;
231 const unsigned int cssIS = stackSizes->cssIS;

```

```

232 // use cssCCTree instead of stackSizes->cssCC and maxCCDepth
233 const unsigned int dssDC = stackSizes->dssDC;
234
235 if(directCallableStackSizeFromTraversal)
236 *directCallableStackSizeFromTraversal = maxDCDepth * dssDC;
237 if(directCallableStackSizeFromState)
238 *directCallableStackSizeFromState = maxDCDepth * dssDC;
239
240 // upper bound on continuation stack used by CH or MS programs including the call tree of
241 // continuation callables
242 unsigned int cssCHorMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
243
244 // clang-format off
245 if(continuationStackSize)
246 *continuationStackSize
247 = cssRG + cssCCTree
248 + (std::max(maxTraceDepth, 1u) - 1) * cssCHorMSPlusCCTree
249 + std::min(maxTraceDepth, 1u) * std::max(cssCHorMSPlusCCTree, cssIS + cssAH);
250 // clang-format on
251
252 return OPTIX_SUCCESS;
253 }
254
255 inline OptixResult optixUtilComputeStackSizesSimplePathTracer(OptixProgramGroup programGroupRG,
256 OptixProgramGroup programGroupMS1,
257 const OptixProgramGroup* programGroupCH1,
258 unsigned int programGroupCH1Count,
259 OptixProgramGroup programGroupMS2,
260 const OptixProgramGroup* programGroupCH2,
261 unsigned int programGroupCH2Count,
262 unsigned int*
263 directCallableStackSizeFromTraversal,
264 unsigned int* directCallableStackSizeFromState,
265 unsigned int* continuationStackSize,
266 OptixPipeline pipeline)
267 {
268 if(!programGroupCH1 && (programGroupCH1Count > 0))
269 return OPTIX_ERROR_INVALID_VALUE;
270 if(!programGroupCH2 && (programGroupCH2Count > 0))
271 return OPTIX_ERROR_INVALID_VALUE;
272
273 OptixResult result;
274
275 OptixStackSizes stackSizesRG = {};
276 result = optixProgramGroupGetStackSize(programGroupRG, &stackSizesRG, pipeline);
277 if(result != OPTIX_SUCCESS)
278 return result;
279
280 OptixStackSizes stackSizesMS1 = {};
281 result = optixProgramGroupGetStackSize(programGroupMS1, &stackSizesMS1,
282 pipeline);
283 if(result != OPTIX_SUCCESS)
284 return result;
285
286 OptixStackSizes stackSizesCH1 = {};
287 for(unsigned int i = 0; i < programGroupCH1Count; ++i)
288 {
289 result = optixUtilAccumulateStackSizes(programGroupCH1[i], &stackSizesCH1, pipeline);
290 if(result != OPTIX_SUCCESS)
291 return result;
292 }
293
294 OptixStackSizes stackSizesMS2 = {};
295 result = optixProgramGroupGetStackSize(programGroupMS2, &stackSizesMS2,
296 pipeline);
297 if(result != OPTIX_SUCCESS)
298 return result;
299
300 return result;
301 }

```

```

311
312 OptixStackSizes stackSizesCH2 = {};
313 memset(&stackSizesCH2, 0, sizeof(OptixStackSizes));
314 for(unsigned int i = 0; i < programGroupCH2Count; ++i)
315 {
316 result = optixUtilAccumulateStackSizes(programGroupCH2[i], &stackSizesCH2, pipeline);
317 if(result != OPTIX_SUCCESS)
318 return result;
319 }
320
321 const unsigned int cssRG = stackSizesRG.cssRG;
322 const unsigned int cssMS1 = stackSizesMS1.cssMS;
323 const unsigned int cssCH1 = stackSizesCH1.cssCH;
324 const unsigned int cssMS2 = stackSizesMS2.cssMS;
325 const unsigned int cssCH2 = stackSizesCH2.cssCH;
326 // no AH, IS, CC, or DC programs
327
328 if(directCallableStackSizeFromTraversal)
329 *directCallableStackSizeFromTraversal = 0;
330 if(directCallableStackSizeFromState)
331 *directCallableStackSizeFromState = 0;
332
333 if(continuationStackSize)
334 *continuationStackSize = cssRG + std::max(cssMS1, cssCH1 + std::max(cssMS2, cssCH2));
335
336 return OPTIX_SUCCESS;
337 }
338 // end group optix_utilities
339
340 #ifdef __cplusplus
341 }
342 #endif
343 #endif // OPTIX_OPTIX_STACK_SIZE_H

```

## 8.23 optix\_stubs.h File Reference

### Macros

- `#define WIN32_LEAN_AND_MEAN 1`

### Functions

- `static void * optixLoadWindowsDllFromName (const char *optixDllName)`
- `static void * optixLoadWindowsDll ()`
- `OPTIXAPI OptixResult optixInitWithHandle (void **handlePtr)`
- `OPTIXAPI OptixResult optixInit (void)`
- `OPTIXAPI OptixResult optixUninitWithHandle (void *handle)`

### Variables

- `OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL`

### 8.23.1 Detailed Description

OptiX public API header.

#### Author

NVIDIA Corporation

## 8.23.2 Macro Definition Documentation

### 8.23.2.1 WIN32\_LEAN\_AND\_MEAN

```
#define WIN32_LEAN_AND_MEAN 1
```

## 8.23.3 Function Documentation

### 8.23.3.1 optixLoadWindowsDll()

```
static void * optixLoadWindowsDll () [static]
```

### 8.23.3.2 optixLoadWindowsDllFromName()

```
static void * optixLoadWindowsDllFromName (
 const char * optixDllName) [static]
```

## 8.24 optix\_stubs.h

[Go to the documentation of this file.](#)

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
16 * contributors may be used to endorse or promote products derived from
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
31
32
33
34
35 #ifndef OPTIX_OPTIX_STUBS_H
36 #define OPTIX_OPTIX_STUBS_H
37
38 #include "optix_function_table.h"
39
40 #ifdef _WIN32
41 #ifndef WIN32_LEAN_AND_MEAN
42 #define WIN32_LEAN_AND_MEAN 1
43 #endif
44 #include <windows.h>
45 // The cfgmgr32 header is necessary for interrogating driver information in the registry.
46 // For convenience the library is also linked in automatically using the #pragma command.
```

```

47 #include <cfgmgr32.h>
48 #pragma comment(lib, "Cfgmgr32.lib")
49 #include <string.h>
50 #else
51 #include <dlfcn.h>
52 #endif
53
54 #ifndef OPTIXAPI
55 # ifdef OPTIX_ENABLE_SDK_MIXING
56 # define OPTIXAPI static
57 # else // OPTIX_ENABLE_SDK_MIXING
58 # ifdef __cplusplus
59 # define OPTIXAPI extern "C"
60 # else // __cplusplus
61 # define OPTIXAPI
62 # endif // __cplusplus
63 # endif // OPTIX_ENABLE_SDK_MIXING
64 #endif // OPTIXAPI
65
66 #ifdef __cplusplus
67 extern "C" {
68 #endif
69
70 // The function table needs to be defined in exactly one translation unit. This can be
71 // achieved by including optix_function_table_definition.h in that translation unit.
72 extern OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL;
73
74 #ifdef __cplusplus
75 }
76 #endif
77
78 #ifdef _WIN32
79 #if defined(_MSC_VER)
80 // Visual Studio produces warnings suggesting strcpy and friends being replaced with _s
81 // variants. All the string lengths and allocation sizes have been calculated and should
82 // be safe, so we are disabling this warning to increase compatibility.
83 #pragma warning(push)
84 #pragma warning(disable : 4996)
85 #endif
86
87 static void* optixLoadWindowsDllFromName(const char* optixDllName)
88 {
89 void* handle = NULL;
90
91 // Try the bare dll name first. This picks it up in the local path, followed by
92 // standard Windows paths.
93 handle = LoadLibraryA((LPSTR)optixDllName);
94 if(handle)
95 return handle;
96 // If we don't find it in the default dll search path, try the system paths
97
98 // Get the size of the path first, then allocate
99 unsigned int size = GetSystemDirectoryA(NULL, 0);
100 if(size == 0)
101 {
102 // Couldn't get the system path size, so bail
103 return NULL;
104 }
105 size_t pathSize = size + 1 + strlen(optixDllName);
106 char* systemPath = (char*)malloc(pathSize);
107 if(systemPath == NULL)
108 return NULL;
109 if(GetSystemDirectoryA(systemPath, size) != size - 1)
110 {
111 // Something went wrong
112 free(systemPath);
113 return NULL;
114 }
115 }

```

```

116 strcat(systemPath, "\\");
117 strcat(systemPath, optixDllName);
118 handle = LoadLibraryA(systemPath);
119 free(systemPath);
120 if(handle)
121 return handle;
122
123 // If we didn't find it, go looking in the register store. Since nvoptix.dll doesn't
124 // have its own registry entry, we are going to look for the opengl driver which lives
125 // next to nvoptix.dll. 0 (null) will be returned if any errors occurred.
126
127 static const char* deviceInstanceIdentifiersGUID = "{4d36e968-e325-11ce-bfc1-08002be10318}";
128 const ULONG flags = CM_GETIDLIST_FILTER_CLASS |
CM_GETIDLIST_FILTER_PRESENT;
129 ULONG deviceListSize = 0;
130 if(CM_Get_Device_ID_List_SizeA(&deviceListSize, deviceInstanceIdentifiersGUID, flags) != CR_SUCCESS)
131 {
132 return NULL;
133 }
134 char* deviceNames = (char*)malloc(deviceListSize);
135 if(deviceNames == NULL)
136 return NULL;
137 if(CM_Get_Device_ID_ListA(deviceInstanceIdentifiersGUID, deviceNames, deviceListSize, flags))
138 {
139 free(deviceNames);
140 return NULL;
141 }
142 DEVINST devID = 0;
143 char* dllPath = NULL;
144
145 // Continue to the next device if errors are encountered.
146 for(char* deviceName = deviceNames; *deviceName; deviceName += strlen(deviceName) + 1)
147 {
148 if(CM_Locate_DevNodeA(&devID, deviceName, CM_LOCATE_DEVNODE_NORMAL) != CR_SUCCESS)
149 {
150 continue;
151 }
152 HKEY regKey = 0;
153 if(CM_Open_DevNode_Key(devID, KEY_QUERY_VALUE, 0, RegDisposition_OpenExisting, ®Key,
CM_REGISTRY_SOFTWARE) != CR_SUCCESS)
154 {
155 continue;
156 }
157 const char* valueName = "OpenGLDriverName";
158 DWORD valueSize = 0;
159 LSTATUS ret = RegQueryValueExA(regKey, valueName, NULL, NULL, NULL, &valueSize);
160 if(ret != ERROR_SUCCESS)
161 {
162 RegCloseKey(regKey);
163 continue;
164 }
165 char* regValue = (char*)malloc(valueSize);
166 if(regValue == NULL)
167 {
168 RegCloseKey(regKey);
169 continue;
170 }
171 ret = RegQueryValueExA(regKey, valueName, NULL, NULL, (LPBYTE)regValue, &valueSize);
172 if(ret != ERROR_SUCCESS)
173 {
174 free(regValue);
175 RegCloseKey(regKey);
176 continue;
177 }
178 // Strip the opengl driver dll name from the string then create a new string with
179 // the path and the nvoptix.dll name
180 for(int i = (int)valueSize - 1; i >= 0 && regValue[i] != '\\'; --i)

```

```

181 regValue[i] = '\0';
182 size_t newPathSize = strlen(regValue) + strlen(optixDllName) + 1;
183 dllPath = (char*)malloc(newPathSize);
184 if(dllPath == NULL)
185 {
186 free(regValue);
187 RegCloseKey(regKey);
188 continue;
189 }
190 strcpy(dllPath, regValue);
191 strcat(dllPath, optixDllName);
192 free(regValue);
193 RegCloseKey(regKey);
194 handle = LoadLibraryA((LPCSTR)dllPath);
195 free(dllPath);
196 if(handle)
197 break;
198 }
199 free(deviceNames);
200 return handle;
201 }
202 #if defined(_MSC_VER)
203 #pragma warning(pop)
204 #endif
205
206 static void* optixLoadWindowsDll()
207 {
208 return optixLoadWindowsDllFromName("nvoptix.dll");
209 }
210 #endif
211
212
213
214
215
216
217
218
219
220
221
222 OPTIXAPI inline OptixResult optixInitWithHandle(void** handlePtr)
223 {
224 // Make sure these functions get initialized to zero in case the DLL and function
225 // table can't be loaded
226 OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorName = 0;
227 OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorString = 0;
228
229 if(!handlePtr)
230 return OPTIX_ERROR_INVALID_VALUE;
231
232 #ifdef _WIN32
233 *handlePtr = optixLoadWindowsDll();
234 if(!*handlePtr)
235 return OPTIX_ERROR_LIBRARY_NOT_FOUND;
236
237 void* symbol = (void*)GetProcAddress((HMODULE)*handlePtr, "optixQueryFunctionTable");
238 if(!symbol)
239 return OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND;
240 #else
241 *handlePtr = dlopen("libnvoptix.so.1", RTLD_NOW);
242 if(!*handlePtr)
243 return OPTIX_ERROR_LIBRARY_NOT_FOUND;
244
245 void* symbol = dlsym(*handlePtr, "optixQueryFunctionTable");
246 if(!symbol)
247 return OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND;
248 #endif
249
250 OptixQueryFunctionTable_t* optixQueryFunctionTable = (OptixQueryFunctionTable_t*)symbol;
251
252 return optixQueryFunctionTable(OPTIX_ABI_VERSION, 0, 0, 0, &OPTIX_FUNCTION_TABLE_SYMBOL,
253 sizeof(OPTIX_FUNCTION_TABLE_SYMBOL));
254 }
255
256
257 OPTIXAPI inline OptixResult optixInit(void)

```

```

261 {
262 void* handle;
263 return optixInitWithHandle(&handle);
264 }
265
271 OPTIXAPI inline OptixResult optixUninitWithHandle(void* handle)
272 {
273 if(!handle)
274 return OPTIX_ERROR_INVALID_VALUE;
275 #ifdef _WIN32
276 if(!FreeLibrary((HMODULE)handle))
277 return OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE;
278 #else
279 if(dlclose(handle))
280 return OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE;
281 #endif
282 OptixFunctionTable empty
283 #ifdef __cplusplus
284 {}
285 #else
286 = { 0 }
287 #endif
288 ;
289 OPTIX_FUNCTION_TABLE_SYMBOL = empty;
290 return OPTIX_SUCCESS;
291 }
292
293 // end group optix_utilities
294
295 #ifndef OPTIX_DOXYGEN_SHOULD_SKIP_THIS
296
297 // Stub functions that forward calls to the corresponding function pointer in the function table.
298
299 OPTIXAPI inline const char* optixGetErrorName(OptixResult result)
300 {
301 if(OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorName)
302 return OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorName(result);
303
304 // If the DLL and symbol table couldn't be loaded, provide a set of error strings
305 // suitable for processing errors related to the DLL loading.
306 switch(result)
307 {
308 case OPTIX_SUCCESS:
309 return "OPTIX_SUCCESS";
310 case OPTIX_ERROR_INVALID_VALUE:
311 return "OPTIX_ERROR_INVALID_VALUE";
312 case OPTIX_ERROR_UNSUPPORTED_ABI_VERSION:
313 return "OPTIX_ERROR_UNSUPPORTED_ABI_VERSION";
314 case OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH:
315 return "OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH";
316 case OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS:
317 return "OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS";
318 case OPTIX_ERROR_LIBRARY_NOT_FOUND:
319 return "OPTIX_ERROR_LIBRARY_NOT_FOUND";
320 case OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND:
321 return "OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND";
322 case OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE:
323 return "OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE";
324 default:
325 return "Unknown OptixResult code";
326 }
327 }
328
329 OPTIXAPI inline const char* optixGetErrorString(OptixResult result)
330 {
331 if(OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorString)
332 return OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorString(result);

```

```

334
335 // If the DLL and symbol table couldn't be loaded, provide a set of error strings
336 // suitable for processing errors related to the DLL loading.
337 switch(result)
338 {
339 case OPTIX_SUCCESS:
340 return "Success";
341 case OPTIX_ERROR_INVALID_VALUE:
342 return "Invalid value";
343 case OPTIX_ERROR_UNSUPPORTED_ABI_VERSION:
344 return "Unsupported ABI version";
345 case OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH:
346 return "Function table size mismatch";
347 case OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS:
348 return "Invalid options to entry function";
349 case OPTIX_ERROR_LIBRARY_NOT_FOUND:
350 return "Library not found";
351 case OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND:
352 return "Entry symbol not found";
353 case OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE:
354 return "Library could not be unloaded";
355 default:
356 return "Unknown OptixResult code";
357 }
358 }
359
360 OPTIXAPI inline OptixResult optixDeviceContextCreate(CUcontext fromContext, const
OptixDeviceContextOptions* options, OptixDeviceContext* context)
361 {
362 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextCreate(fromContext, options, context);
363 }
364
365 OPTIXAPI inline OptixResult optixDeviceContextDestroy(OptixDeviceContext context)
366 {
367 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextDestroy(context);
368 }
369
370 OPTIXAPI inline OptixResult optixDeviceContextGetProperty(OptixDeviceContext context,
OptixDeviceProperty property, void* value, size_t sizeInBytes)
371 {
372 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetProperty(context, property, value,
sizeInBytes);
373 }
374
375 OPTIXAPI inline OptixResult optixDeviceContextSetLogCallback(OptixDeviceContext context,
OptixLogCallback callbackFunction,
376 void* callbackData,
377 unsigned int callbackLevel)
378 {
379 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetLogCallback(context, callbackFunction,
callbackData, callbackLevel);
380 }
381 }
382
383 OPTIXAPI inline OptixResult optixDeviceContextSetCacheEnabled(OptixDeviceContext context, int enabled)
384 {
385 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetCacheEnabled(context, enabled);
386 }
387
388 OPTIXAPI inline OptixResult optixDeviceContextSetCacheLocation(OptixDeviceContext context, const char*
location)
389 {
390 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetCacheLocation(context, location);
391 }
392
393 OPTIXAPI inline OptixResult optixDeviceContextSetCacheDatabaseSizes(OptixDeviceContext context, size_t
lowWaterMark, size_t highWaterMark)
394 {

```

```

395 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetCacheDatabaseSizes(context, lowWaterMark,
highWaterMark);
396 }
397
398 OPTIXAPI inline OptixResult optixDeviceContextGetCacheEnabled(OptixDeviceContext context, int* enabled)
399 {
400 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetCacheEnabled(context, enabled);
401 }
402
403 OPTIXAPI inline OptixResult optixDeviceContextGetCacheLocation(OptixDeviceContext context, char*
location, size_t locationSize)
404 {
405 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetCacheLocation(context, location,
locationSize);
406 }
407
408 OPTIXAPI inline OptixResult optixDeviceContextGetCacheDatabaseSizes(OptixDeviceContext context, size_t*
lowWaterMark, size_t* highWaterMark)
409 {
410 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetCacheDatabaseSizes(context, lowWaterMark,
highWaterMark);
411 }
412
413 OPTIXAPI inline OptixResult optixModuleCreate(OptixDeviceContext context,
414 const OptixModuleCompileOptions* moduleCompileOptions,
415 const OptixPipelineCompileOptions* pipelineCompileOptions,
416 const char* input,
417 size_t inputSize,
418 char* logString,
419 size_t* logStringSize,
420 OptixModule* module)
421 {
422 return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleCreate(context, moduleCompileOptions,
pipelineCompileOptions, input,
423 inputSize, logString, logStringSize, module);
424 }
425
426 OPTIXAPI inline OptixResult optixModuleCreateWithTasks(OptixDeviceContext context,
427 const OptixModuleCompileOptions*
moduleCompileOptions,
428 const OptixPipelineCompileOptions*
pipelineCompileOptions,
429 const char* input,
430 size_t inputSize,
431 char* logString,
432 size_t* logStringSize,
433 OptixModule* module,
434 OptixTask* firstTask)
435 {
436 return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleCreateWithTasks(context, moduleCompileOptions,
pipelineCompileOptions, input,
437 inputSize, logString, logStringSize,
module, firstTask);
438 }
439
440 OPTIXAPI inline OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState*
state)
441 {
442 return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleGetCompilationState(module, state);
443 }
444
445 OPTIXAPI inline OptixResult optixModuleDestroy(OptixModule module)
446 {
447 return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleDestroy(module);
448 }
449
450 OPTIXAPI inline OptixResult optixBuiltinISModuleGet(OptixDeviceContext context,

```

```

451 const OptixModuleCompileOptions* moduleCompileOptions,
452 const OptixPipelineCompileOptions* pipelineCompileOptions,
453 const OptixBuiltinISOOptions* builtinISOOptions,
454 OptixModule* builtinModule)
455 {
456 return OPTIX_FUNCTION_TABLE_SYMBOL.optixBuiltinISModuleGet(context, moduleCompileOptions,
457 pipelineCompileOptions,
458 builtinISOOptions, builtinModule);
459 }
460 OPTIXAPI inline OptixResult optixTaskExecute(OptixTask task,
461 OptixTask* additionalTasks,
462 unsigned int maxNumAdditionalTasks,
463 unsigned int* numAdditionalTasksCreated)
464 {
465 return OPTIX_FUNCTION_TABLE_SYMBOL.optixTaskExecute(task, additionalTasks, maxNumAdditionalTasks,
466 numAdditionalTasksCreated);
467 }
468 OPTIXAPI inline OptixResult optixProgramGroupCreate(OptixDeviceContext context,
469 const OptixProgramGroupDesc* programDescriptions,
470 unsigned int numProgramGroups,
471 const OptixProgramGroupOptions* options,
472 char* logString,
473 size_t* logStringSize,
474 OptixProgramGroup* programGroups)
475 {
476 return OPTIX_FUNCTION_TABLE_SYMBOL.optixProgramGroupCreate(context, programDescriptions,
477 numProgramGroups, options,
478 logString, logStringSize, programGroups);
479 }
480 OPTIXAPI inline OptixResult optixProgramGroupDestroy(OptixProgramGroup programGroup)
481 {
482 return OPTIX_FUNCTION_TABLE_SYMBOL.optixProgramGroupDestroy(programGroup);
483 }
484
485 OPTIXAPI inline OptixResult optixProgramGroupGetStackSize(OptixProgramGroup programGroup,
486 OptixStackSizes* stackSizes, OptixPipeline pipeline)
487 {
488 return OPTIX_FUNCTION_TABLE_SYMBOL.optixProgramGroupGetStackSize(programGroup, stackSizes, pipeline);
489 }
490 OPTIXAPI inline OptixResult optixPipelineCreate(OptixDeviceContext context,
491 const OptixPipelineCompileOptions* pipelineCompileOptions,
492 const OptixPipelineLinkOptions* pipelineLinkOptions,
493 const OptixProgramGroup* programGroups,
494 unsigned int numProgramGroups,
495 char* logString,
496 size_t* logStringSize,
497 OptixPipeline* pipeline)
498 {
499 return OPTIX_FUNCTION_TABLE_SYMBOL.optixPipelineCreate(context, pipelineCompileOptions,
500 pipelineLinkOptions, programGroups,
501 numProgramGroups, logString, logStringSize,
502 pipeline);
503 }
504 OPTIXAPI inline OptixResult optixPipelineDestroy(OptixPipeline pipeline)
505 {
506 return OPTIX_FUNCTION_TABLE_SYMBOL.optixPipelineDestroy(pipeline);
507 }
508 OPTIXAPI inline OptixResult optixPipelineSetStackSize(OptixPipeline pipeline,
509 unsigned int directCallableStackSizeFromTraversal,
510 unsigned int directCallableStackSizeFromState,
511 unsigned int continuationStackSize,

```

```

512 unsigned int maxTraversableGraphDepth)
513 {
514 return OPTIX_FUNCTION_TABLE_SYMBOL.optixPipelineSetStackSize(pipeline,
directCallableStackSizeFromTraversal,
515 directCallableStackSizeFromState,
516 continuationStackSize,
maxTraversableGraphDepth);
517 }
518
519 OPTIXAPI inline OptixResult optixAccelComputeMemoryUsage(OptixDeviceContext context,
520 const OptixAccelBuildOptions* accelOptions,
521 const OptixBuildInput* buildInputs,
522 unsigned int numBuildInputs,
523 OptixAccelBufferSizes* bufferSizes)
524 {
525 return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelComputeMemoryUsage(context, accelOptions, buildInputs,
numBuildInputs, bufferSizes);
526 }
527
528 OPTIXAPI inline OptixResult optixAccelBuild(OptixDeviceContext context,
529 CUstream stream,
530 const OptixAccelBuildOptions* accelOptions,
531 const OptixBuildInput* buildInputs,
532 unsigned int numBuildInputs,
533 CUdeviceptr tempBuffer,
534 size_t tempBufferSizeInBytes,
535 CUdeviceptr outputBuffer,
536 size_t outputBufferSizeInBytes,
537 OptixTraversableHandle* outputHandle,
538 const OptixAccelEmitDesc* emittedProperties,
539 unsigned int numEmittedProperties)
540 {
541 return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelBuild(context, stream, accelOptions, buildInputs,
numBuildInputs, tempBuffer,
542 tempBufferSizeInBytes, outputBuffer,
outputBufferSizeInBytes,
543 outputHandle, emittedProperties, numEmittedProperties);
544 }
545
546
547 OPTIXAPI inline OptixResult optixAccelGetRelocationInfo(OptixDeviceContext context,
OptixTraversableHandle handle, OptixRelocationInfo* info)
548 {
549 return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelGetRelocationInfo(context, handle, info);
550 }
551
552
553 OPTIXAPI inline OptixResult optixCheckRelocationCompatibility(OptixDeviceContext context, const
OptixRelocationInfo* info, int* compatible)
554 {
555 return OPTIX_FUNCTION_TABLE_SYMBOL.optixCheckRelocationCompatibility(context, info, compatible);
556 }
557
558 OPTIXAPI inline OptixResult optixAccelRelocate(OptixDeviceContext context,
559 CUstream stream,
560 const OptixRelocationInfo* info,
561 const OptixRelocateInput* relocateInputs,
562 size_t numRelocateInputs,
563 CUdeviceptr targetAccel,
564 size_t targetAccelSizeInBytes,
565 OptixTraversableHandle* targetHandle)
566 {
567 return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelRelocate(context, stream, info, relocateInputs,
numRelocateInputs,
568 targetAccel, targetAccelSizeInBytes, targetHandle);
569 }
570

```

```

571 OPTIXAPI inline OptixResult optixAccelCompact(OptixDeviceContext context,
572 CUstream stream,
573 OptixTraversableHandle inputHandle,
574 CUdeviceptr outputBuffer,
575 size_t outputBufferSizeInBytes,
576 OptixTraversableHandle* outputHandle)
577 {
578 return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelCompact(context, stream, inputHandle, outputBuffer,
579 outputBufferSizeInBytes, outputHandle);
580 }
581
582 OPTIXAPI inline OptixResult optixAccelEmitProperty(OptixDeviceContext context,
583 CUstream stream,
584 OptixTraversableHandle handle,
585 const OptixAccelEmitDesc* emittedProperty)
586 {
587 return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelEmitProperty(context, stream, handle, emittedProperty);
588 }
589
590 OPTIXAPI inline OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext onDevice,
591 CUdeviceptr pointer,
592 OptixTraversableType traversableType,
593 OptixTraversableHandle* traversableHandle)
594 {
595 return OPTIX_FUNCTION_TABLE_SYMBOL.optixConvertPointerToTraversableHandle(onDevice, pointer,
traversableType, traversableHandle);
596 }
597
598 OPTIXAPI inline OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext context,
599 const
OptixOpacityMicromapArrayBuildInput* buildInput,
600 OptixMicromapBufferSizes* bufferSizes)
601 {
602 return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayComputeMemoryUsage(context, buildInput,
bufferSizes);
603 }
604
605 OPTIXAPI inline OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext
context,
606 CUstream stream,
607 const OptixOpacityMicromapArrayBuildInput*
buildInput,
608 const OptixMicromapBuffers* buffers)
609 {
610 return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayBuild(context, stream, buildInput,
buffers);
611 }
612
613 OPTIXAPI inline OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext context,
614 CUdeviceptr opacityMicromapArray,
615 OptixRelocationInfo* info)
616 {
617 return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayGetRelocationInfo(context,
opacityMicromapArray, info);
618 }
619
620 OPTIXAPI inline OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext context,
621 CUstream stream,
622 const OptixRelocationInfo* info,
623 CUdeviceptr
targetOpacityMicromapArray,
624 size_t targetOpacityMicromapArraySizeInBytes)
625 {
626 return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayRelocate(context, stream, info,
targetOpacityMicromapArray,
627 targetOpacityMicromapArraySizeInBytes);
628 }

```

```

629
630 OPTIXAPI inline OptixResult optixDisplacementMicromapArrayComputeMemoryUsage(OptixDeviceContext context,
631 const
OptixDisplacementMicromapArrayBuildInput* buildInput,
632 OptixMicromapBufferSizes*
bufferSizes)
633 {
634 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDisplacementMicromapArrayComputeMemoryUsage(context,
buildInput, bufferSizes);
635 }
636
637 OPTIXAPI inline OptixResult optixDisplacementMicromapArrayBuild(OptixDeviceContext context,
638 CUstream stream,
639 const
OptixDisplacementMicromapArrayBuildInput* buildInput,
640 const OptixMicromapBuffers* buffers)
641 {
642 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDisplacementMicromapArrayBuild(context, stream, buildInput,
buffers);
643 }
644
645 OPTIXAPI inline OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer)
646 {
647 return OPTIX_FUNCTION_TABLE_SYMBOL.optixSbtRecordPackHeader(programGroup,
sbtRecordHeaderHostPointer);
648 }
649
650 OPTIXAPI inline OptixResult optixLaunch(OptixPipeline pipeline,
651 CUstream stream,
652 CUdeviceptr pipelineParams,
653 size_t pipelineParamsSize,
654 const OptixShaderBindingTable* sbt,
655 unsigned int width,
656 unsigned int height,
657 unsigned int depth)
658 {
659 return OPTIX_FUNCTION_TABLE_SYMBOL.optixLaunch(pipeline, stream, pipelineParams, pipelineParamsSize,
sbt, width, height, depth);
660 }
661
662 OPTIXAPI inline OptixResult optixDenoiserCreate(OptixDeviceContext context,
663 OptixDenoiserModelKind modelKind,
664 const OptixDenoiserOptions* options,
665 OptixDenoiser* returnHandle)
666 {
667 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserCreate(context, modelKind, options, returnHandle);
668 }
669
670 OPTIXAPI inline OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context,
671 const void* data,
672 size_t dataSizeInBytes,
673 OptixDenoiser* returnHandle)
674 {
675 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserCreateWithUserModel(context, data, dataSizeInBytes,
returnHandle);
676 }
677
678 OPTIXAPI inline OptixResult optixDenoiserDestroy(OptixDenoiser handle)
679 {
680 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserDestroy(handle);
681 }
682
683 OPTIXAPI inline OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser handle,
684 unsigned int maximumInputWidth,
685 unsigned int maximumInputHeight,
686 OptixDenoiserSizes* returnSizes)

```

```

687 {
688 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserComputeMemoryResources(handle, maximumInputWidth,
maximumInputHeight, returnSizes);
689 }
690
691 OPTIXAPI inline OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
692 CUstream stream,
693 unsigned int inputWidth,
694 unsigned int inputHeight,
695 CUdeviceptr denoiserState,
696 size_t denoiserStateSizeInBytes,
697 CUdeviceptr scratch,
698 size_t scratchSizeInBytes)
699 {
700 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserSetup(denoiser, stream, inputWidth, inputHeight,
denoiserState,
701 denoiserStateSizeInBytes, scratch,
scratchSizeInBytes);
702 }
703
704 OPTIXAPI inline OptixResult optixDenoiserInvoke(OptixDenoiser handle,
705 CUstream stream,
706 const OptixDenoiserParams* params,
707 CUdeviceptr denoiserData,
708 size_t denoiserDataSize,
709 const OptixDenoiserGuideLayer* guideLayer,
710 const OptixDenoiserLayer* layers,
711 unsigned int numLayers,
712 unsigned int inputOffsetX,
713 unsigned int inputOffsetY,
714 CUdeviceptr scratch,
715 size_t scratchSizeInBytes)
716 {
717 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserInvoke(handle, stream, params, denoiserData,
denoiserDataSize,
718 guideLayer, layers, numLayers, inputOffsetX,
inputOffsetY,
719 scratch, scratchSizeInBytes);
720 }
721
722 OPTIXAPI inline OptixResult optixDenoiserComputeIntensity(OptixDenoiser handle,
723 CUstream stream,
724 const OptixImage2D* inputImage,
725 CUdeviceptr outputIntensity,
726 CUdeviceptr scratch,
727 size_t scratchSizeInBytes)
728 {
729 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserComputeIntensity(handle, stream, inputImage,
outputIntensity,
730 scratch, scratchSizeInBytes);
731 }
732
733 OPTIXAPI inline OptixResult optixDenoiserComputeAverageColor(OptixDenoiser handle,
734 CUstream stream,
735 const OptixImage2D* inputImage,
736 CUdeviceptr outputAverageColor,
737 CUdeviceptr scratch,
738 size_t scratchSizeInBytes)
739 {
740 return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserComputeAverageColor(handle, stream, inputImage,
outputAverageColor,
741 scratch, scratchSizeInBytes);
742 }
743
744 #endif // OPTIX_DOXYGEN_SHOULD_SKIP_THIS
745
746 #endif // OPTIX_OPTIX_STUBS_H

```

## 8.25 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 OptixDevicePropertyShaderExecutionReorderingFlags OptixDevicePropertyShaderExecutionReorderingFlags
- 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 enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
- typedef struct OptixDenoiserOptions OptixDenoiserOptions
- typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
- typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
- typedef struct OptixDenoiserLayer OptixDenoiserLayer
- 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 ,
OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING = 0x200A }
• enum OptixDeviceContextValidationMode {
 OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0 ,
 OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF }
• enum OptixDevicePropertyShaderExecutionReorderingFlags {
 OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_NONE = 0 ,
 OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD = 1
 << 0 }
• 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_BYTE3 = 0x2101 ,
 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 OptixDenoiserAlphaMode {

```

OPTIX_DENOISER_ALPHA_MODE_COPY = 0 ,
OPTIX_DENOISER_ALPHA_MODE_DENOISE = 1 }

```
  - 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 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 }

```
 - 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 }

```
 - enum OptixQueryFunctionTableOptions { OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0 }

8.25.1 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`

8.26 optix_types.h

[Go to the documentation of this file.](#)

```

1
2 /*
3 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
4 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
5 *
6 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
7 * property and proprietary rights in and to this material, related
8 * documentation and any modifications thereto. Any use, reproduction,
9 * disclosure or distribution of this material and related documentation
10 * without an express license agreement from NVIDIA CORPORATION or
11 * its affiliates is strictly prohibited.
12 */
13
14
15
16
17
18
19
20 #ifndef OPTIX_OPTIX_TYPES_H
21 #define OPTIX_OPTIX_TYPES_H
22
23 #if !defined(__CUDACC_RTC__)
24 #include <stddef.h> /* for size_t */
25 #endif
26
27 #ifdef NV_MODULE_OPTIX
28 // This is a mechanism to include <g_nvconfig.h> in driver builds only and translate any nvconfig macro to
29 // a custom OPTIX-specific macro, that can also be used in SDK builds/installs
30 #include <exp/misc/optix_nvconfig_translate.h> // includes <g_nvconfig.h>
31 #endif // NV_MODULE_OPTIX
32
33
34
35
36
37
38
39 // This typedef should match the one in cuda.h in order to avoid compilation errors.
40 #if defined(_WIN64) || defined(__LP64__)
41 typedef unsigned long long CUdeviceptr;
42 #else
43 typedef unsigned int CUdeviceptr;
44 #endif
45
46
47 #endif
48
49
50 typedef struct OptixDeviceContext_t* OptixDeviceContext;
51
52
53 typedef struct OptixModule_t* OptixModule;
54
55
56 typedef struct OptixProgramGroup_t* OptixProgramGroup;
57
58
59 typedef struct OptixPipeline_t* OptixPipeline;
60
61
62 typedef struct OptixDenoiser_t* OptixDenoiser;
63
64
65 typedef struct OptixTask_t* OptixTask;
66
67
68 typedef unsigned long long OptixTraversableHandle;
69
70
71 typedef unsigned int OptixVisibilityMask;
72

```

```

74 #define OPTIX_SBT_RECORD_HEADER_SIZE ((size_t)32)
75
77 #define OPTIX_SBT_RECORD_ALIGNMENT 16ull
78
80 #define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128ull
81
83 #define OPTIX_INSTANCE_BYTE_ALIGNMENT 16ull
84
86 #define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8ull
87
89 #define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16ull
90
92 #define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
93
95 #define OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
96
98 #define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
99
101 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
102
104 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
105
108 #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT (0)
109 #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE (1)
110 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT (2)
111 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE (3)
112
115 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT (-1)
116 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE (-2)
117 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT (-3)
118 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE (-4)
119
121 #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
122
124 #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
125
127 #define OPTIX_DISPLACEMENT_MICROMAP_MAX_SUBDIVISION_LEVEL 5
128
130 #define OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
131
133 #define OPTIX_DISPLACEMENT_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
134
142 typedef enum OptixResult
143 {
144     OPTIX_SUCCESS = 0,
145     OPTIX_ERROR_INVALID_VALUE = 7001,
146     OPTIX_ERROR_HOST_OUT_OF_MEMORY = 7002,
147     OPTIX_ERROR_INVALID_OPERATION = 7003,
148     OPTIX_ERROR_FILE_IO_ERROR = 7004,
149     OPTIX_ERROR_INVALID_FILE_FORMAT = 7005,
150     OPTIX_ERROR_DISK_CACHE_INVALID_PATH = 7010,
151     OPTIX_ERROR_DISK_CACHE_PERMISSION_ERROR = 7011,
152     OPTIX_ERROR_DISK_CACHE_DATABASE_ERROR = 7012,
153     OPTIX_ERROR_DISK_CACHE_INVALID_DATA = 7013,
154     OPTIX_ERROR_LAUNCH_FAILURE = 7050,
155     OPTIX_ERROR_INVALID_DEVICE_CONTEXT = 7051,
156     OPTIX_ERROR_CUDA_NOT_INITIALIZED = 7052,
157     OPTIX_ERROR_VALIDATION_FAILURE = 7053,
158     OPTIX_ERROR_INVALID_INPUT = 7200,
159     OPTIX_ERROR_INVALID_LAUNCH_PARAMETER = 7201,
160     OPTIX_ERROR_INVALID_PAYLOAD_ACCESS = 7202,
161     OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS = 7203,
162     OPTIX_ERROR_INVALID_FUNCTION_USE = 7204,
163     OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS = 7205,
164     OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250,
165     OPTIX_ERROR_PIPELINE_LINK_ERROR = 7251,
166     OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE = 7270,

```

```

167     OPTIX_ERROR_INTERNAL_COMPILER_ERROR           = 7299,
168     OPTIX_ERROR_DENOISER_MODEL_NOT_SET           = 7300,
169     OPTIX_ERROR_DENOISER_NOT_INITIALIZED         = 7301,
170     OPTIX_ERROR_NOT_COMPATIBLE                   = 7400,
171     OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH           = 7500,
172     OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED   = 7501,
173     OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID          = 7502,
174     OPTIX_ERROR_NOT_SUPPORTED                     = 7800,
175     OPTIX_ERROR_UNSUPPORTED_ABI_VERSION          = 7801,
176     OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH     = 7802,
177     OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS    = 7803,
178     OPTIX_ERROR_LIBRARY_NOT_FOUND                = 7804,
179     OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND           = 7805,
180     OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE           = 7806,
181     OPTIX_ERROR_DEVICE_OUT_OF_MEMORY             = 7807,
182     OPTIX_ERROR_CUDA_ERROR                       = 7900,
183     OPTIX_ERROR_INTERNAL_ERROR                   = 7990,
184     OPTIX_ERROR_UNKNOWN                          = 7999,
185 } OptixResult;
186
187
188
189 typedef enum OptixDeviceProperty
190 {
191     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001,
192
193     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002,
194
195     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003,
196
197     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004,
198
199     OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005,
200
201     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006,
202
203     OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007,
204
205     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008,
206
207     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009,
208
209     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING = 0x200A,
210 } OptixDeviceProperty;
211
212
213
214
215 typedef void (*OptixLogCallback)(unsigned int level, const char* tag, const char* message, void* cbdata);
216
217
218
219 typedef enum OptixDeviceContextValidationMode
220 {
221     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0,
222     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF
223 } OptixDeviceContextValidationMode;
224
225
226
227
228 typedef struct OptixDeviceContextOptions
229 {
230     OptixLogCallback logCallbackFunction;
231     void* logCallbackData;
232     int logCallbackLevel;
233     OptixDeviceContextValidationMode validationMode;
234 } OptixDeviceContextOptions;
235
236
237
238
239 typedef enum OptixDevicePropertyShaderExecutionReorderingFlags
240 {
241     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_NONE = 0,
242
243     // Standard thread reordering is supported
244     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD = 1 << 0,
245 } OptixDevicePropertyShaderExecutionReorderingFlags;
246
247

```

```

304 typedef enum OptixGeometryFlags
305 {
307     OPTIX_GEOMETRY_FLAG_NONE = 0,
308
311     OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u « 0,
312
316     OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u « 1,
317
321     OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u « 2,
322 } OptixGeometryFlags;
323
329 typedef enum OptixHitKind
330 {
332     OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE,
334     OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF
335 } OptixHitKind;
336
338 typedef enum OptixIndicesFormat
339 {
341     OPTIX_INDICES_FORMAT_NONE = 0,
343     OPTIX_INDICES_FORMAT_UNSIGNED_BYTE3 = 0x2101,
345     OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102,
347     OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103
348 } OptixIndicesFormat;
349
351 typedef enum OptixVertexFormat
352 {
353     OPTIX_VERTEX_FORMAT_NONE          = 0,
354     OPTIX_VERTEX_FORMAT_FLOAT3        = 0x2121,
355     OPTIX_VERTEX_FORMAT_FLOAT2        = 0x2122,
356     OPTIX_VERTEX_FORMAT_HALF3        = 0x2123,
357     OPTIX_VERTEX_FORMAT_HALF2        = 0x2124,
358     OPTIX_VERTEX_FORMAT_SNORM16_3    = 0x2125,
359     OPTIX_VERTEX_FORMAT_SNORM16_2    = 0x2126
360 } OptixVertexFormat;
361
363 typedef enum OptixTransformFormat
364 {
365     OPTIX_TRANSFORM_FORMAT_NONE          = 0,
366     OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1,
367 } OptixTransformFormat;
368
369 typedef enum OptixDisplacementMicromapBiasAndScaleFormat
370 {
371     OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_NONE = 0,
372     OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_FLOAT2 = 0x2241,
373     OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_HALF2 = 0x2242,
374 } OptixDisplacementMicromapBiasAndScaleFormat;
375
376 typedef enum OptixDisplacementMicromapDirectionFormat
377 {
378     OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_NONE = 0,
379     OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_FLOAT3 = 0x2261,
380     OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_HALF3 = 0x2262,
381 } OptixDisplacementMicromapDirectionFormat;
382
384 typedef enum OptixOpacityMicromapFormat
385 {
387     OPTIX_OPACITY_MICROMAP_FORMAT_NONE = 0,
389     OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE = 1,
391     OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE = 2,
392 } OptixOpacityMicromapFormat;
393
395 typedef enum OptixOpacityMicromapArrayIndexingMode
396 {
398     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0,
401     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1,

```

```

405     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2,
406 } OptixOpacityMicromapArrayIndexingMode;
407
412 typedef struct OptixOpacityMicromapUsageCount
413 {
414     unsigned int count;
415     unsigned int subdivisionLevel;
416     OptixOpacityMicromapFormat format;
417 } OptixOpacityMicromapUsageCount;
418
423 typedef struct OptixBuildInputOpacityMicromap
424 {
425     OptixOpacityMicromapArrayIndexingMode indexingMode;
426
427     CUdeviceptr opacityMicromapArray;
428
429     CUdeviceptr indexBuffer;
430
431     unsigned int indexSizeInBytes;
432
433     unsigned int indexStrideInBytes;
434
435     unsigned int indexOffset;
436
437     unsigned int numMicromapUsageCounts;
438     const OptixOpacityMicromapUsageCount* micromapUsageCounts;
439 } OptixBuildInputOpacityMicromap;
440
446 typedef struct OptixRelocateInputOpacityMicromap
447 {
448     CUdeviceptr opacityMicromapArray;
449 } OptixRelocateInputOpacityMicromap;
450
456 typedef enum OptixDisplacementMicromapFormat
457 {
458     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_NONE = 0,
459     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_64_MICRO_TRIS_64_BYTES = 1,
460     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_256_MICRO_TRIS_128_BYTES = 2,
461     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_1024_MICRO_TRIS_128_BYTES = 3,
462 } OptixDisplacementMicromapFormat;
463
469 typedef enum OptixDisplacementMicromapFlags
470 {
471     OPTIX_DISPLACEMENT_MICROMAP_FLAG_NONE = 0,
472
473     OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 << 0,
474
475     OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 << 1,
476 } OptixDisplacementMicromapFlags;
477
478 typedef enum OptixDisplacementMicromapTriangleFlags
479 {
480     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_NONE = 0,
481     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_01 = 1 << 0,
482     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_12 = 1 << 1,
483     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_20 = 1 << 2,
484 } OptixDisplacementMicromapTriangleFlags;
485
486 typedef struct OptixDisplacementMicromapDesc
487 {
488     unsigned int byteOffset;
489     unsigned short subdivisionLevel;
490     unsigned short format;
491 } OptixDisplacementMicromapDesc;
492

```

```

520 typedef struct OptixDisplacementMicromapHistogramEntry
521 {
522     unsigned int          count;
523     unsigned int          subdivisionLevel;
524     OptixDisplacementMicromapFormat format;
525 } OptixDisplacementMicromapHistogramEntry;
526
527
528
529
530 typedef struct OptixDisplacementMicromapArrayBuildInput
531 {
532     OptixDisplacementMicromapFlags          flags;
533     CUdeviceptr displacementValuesBuffer;
534     CUdeviceptr perDisplacementMicromapDescBuffer;
535     unsigned int perDisplacementMicromapDescStrideInBytes;
536     unsigned int numDisplacementMicromapHistogramEntries;
537     const OptixDisplacementMicromapHistogramEntry* displacementMicromapHistogramEntries;
538 } OptixDisplacementMicromapArrayBuildInput;
539
540
541
542
543 typedef struct OptixDisplacementMicromapUsageCount
544 {
545     unsigned int          count;
546     unsigned int          subdivisionLevel;
547     OptixDisplacementMicromapFormat format;
548 } OptixDisplacementMicromapUsageCount;
549
550
551
552
553 typedef enum OptixDisplacementMicromapArrayIndexingMode
554 {
555     OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0,
556     OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1,
557     OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2,
558 } OptixDisplacementMicromapArrayIndexingMode;
559
560
561
562
563 typedef struct OptixBuildInputDisplacementMicromap
564 {
565     OptixDisplacementMicromapArrayIndexingMode indexingMode;
566
567     CUdeviceptr displacementMicromapArray;
568     CUdeviceptr displacementMicromapIndexBuffer;
569     CUdeviceptr vertexDirectionsBuffer;
570     CUdeviceptr vertexBiasAndScaleBuffer;
571     CUdeviceptr triangleFlagsBuffer;
572
573     unsigned int displacementMicromapIndexOffset;
574     unsigned int displacementMicromapIndexStrideInBytes;
575     unsigned int displacementMicromapIndexSizeInBytes;
576
577     OptixDisplacementMicromapDirectionFormat vertexDirectionFormat;
578     unsigned int vertexDirectionStrideInBytes;
579
580     OptixDisplacementMicromapBiasAndScaleFormat vertexBiasAndScaleFormat;
581     unsigned int vertexBiasAndScaleStrideInBytes;
582
583     unsigned int triangleFlagsStrideInBytes;
584
585     unsigned int numDisplacementMicromapUsageCounts;
586     const OptixDisplacementMicromapUsageCount* displacementMicromapUsageCounts;
587 } OptixBuildInputDisplacementMicromap;
588
589
590
591
592 typedef struct OptixBuildInputTriangleArray
593 {
594     const CUdeviceptr* vertexBuffers;
595
596     unsigned int numVertices;
597
598     OptixVertexFormat vertexFormat;
599 } OptixBuildInputTriangleArray;

```

```

649     unsigned int vertexStrideInBytes;
650
654     CUdeviceptr indexBuffer;
655
657     unsigned int numIndexTriplets;
658
660     OptixIndicesFormat indexFormat;
661
664     unsigned int indexStrideInBytes;
665
669     CUdeviceptr preTransform;
670
674     const unsigned int* flags;
675
677     unsigned int numSbtRecords;
678
682     CUdeviceptr sbtIndexOffsetBuffer;
683
685     unsigned int sbtIndexOffsetSizeInBytes;
686
689     unsigned int sbtIndexOffsetStrideInBytes;
690
693     unsigned int primitiveIndexOffset;
694
696     OptixTransformFormat transformFormat;
697
699     OptixBuildInputOpacityMicromap opacityMicromap;
701     OptixBuildInputDisplacementMicromap displacementMicromap;
702
703 } OptixBuildInputTriangleArray;
704
708 typedef struct OptixRelocateInputTriangleArray
709 {
712     unsigned int numSbtRecords;
713
715     OptixRelocateInputOpacityMicromap opacityMicromap;
716 } OptixRelocateInputTriangleArray;
717
720 typedef enum OptixPrimitiveType
721 {
723     OPTIX_PRIMITIVE_TYPE_CUSTOM = 0x2500,
725     OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE = 0x2501,
727     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE = 0x2502,
729     OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR = 0x2503,
731     OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM = 0x2504,
733     OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE = 0x2505,
735     OPTIX_PRIMITIVE_TYPE_SPHERE = 0x2506,
737     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER = 0x2507,
739     OPTIX_PRIMITIVE_TYPE_TRIANGLE = 0x2531,
741     OPTIX_PRIMITIVE_TYPE_DISPLACED_MICROMESH_TRIANGLE = 0x2532,
742 } OptixPrimitiveType;
743
747 typedef enum OptixPrimitiveTypeFlags
748 {
750     OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM = 1 << 0,
752     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE = 1 << 1,
754     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE = 1 << 2,
756     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR = 1 << 3,
758     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM = 1 << 4,
760     OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE = 1 << 5,
762     OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE = 1 << 6,
764     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER = 1 << 7,
766     OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE = 1 << 31,
768     OPTIX_PRIMITIVE_TYPE_FLAGS_DISPLACED_MICROMESH_TRIANGLE = 1 << 30,
769 } OptixPrimitiveTypeFlags;
770
773 typedef enum OptixCurveEndcapFlags

```

```

774 {
775     OPTIX_CURVE_ENDCAP_DEFAULT           = 0,
776     OPTIX_CURVE_ENDCAP_ON              = 1 « 0,
777 } OptixCurveEndcapFlags;
778
779 typedef struct OptixBuildInputCurveArray
780 {
781     OptixPrimitiveType curveType;
782     unsigned int numPrimitives;
783
784     const CUdeviceptr* vertexBuffers;
785     unsigned int numVertices;
786     unsigned int vertexStrideInBytes;
787
788     const CUdeviceptr* widthBuffers;
789     unsigned int widthStrideInBytes;
790
791     const CUdeviceptr* normalBuffers;
792     unsigned int normalStrideInBytes;
793
794     CUdeviceptr indexBuffer;
795     unsigned int indexStrideInBytes;
796
797     unsigned int flag;
798
799     unsigned int primitiveIndexOffset;
800
801     unsigned int endcapFlags;
802 } OptixBuildInputCurveArray;
803
804 typedef struct OptixBuildInputSphereArray
805 {
806     const CUdeviceptr* vertexBuffers;
807     unsigned int vertexStrideInBytes;
808     unsigned int numVertices;
809
810     const CUdeviceptr* radiusBuffers;
811     unsigned int radiusStrideInBytes;
812     int singleRadius;
813
814     const unsigned int* flags;
815
816     unsigned int numSbtRecords;
817     CUdeviceptr sbtIndexOffsetBuffer;
818     unsigned int sbtIndexOffsetSizeInBytes;
819     unsigned int sbtIndexOffsetStrideInBytes;
820
821     unsigned int primitiveIndexOffset;
822 } OptixBuildInputSphereArray;
823
824 typedef struct OptixAabb
825 {
826     float minX;
827     float minY;
828     float minZ;
829     float maxX;
830     float maxY;
831     float maxZ;
832 } OptixAabb;
833
834 typedef struct OptixBuildInputCustomPrimitiveArray
835 {
836     const CUdeviceptr* aabbBuffers;
837
838     unsigned int numPrimitives;
839 }

```

```

938     unsigned int strideInBytes;
939
943     const unsigned int* flags;
944
946     unsigned int numSbtRecords;
947
951     CUdeviceptr sbtIndexOffsetBuffer;
952
954     unsigned int sbtIndexOffsetSizeInBytes;
955
958     unsigned int sbtIndexOffsetStrideInBytes;
959
962     unsigned int primitiveIndexOffset;
963 } OptixBuildInputCustomPrimitiveArray;
964
968 typedef struct OptixBuildInputInstanceArray
969 {
977     CUdeviceptr instances;
978
980     unsigned int numInstances;
981
985     unsigned int instanceStride;
986 } OptixBuildInputInstanceArray;
987
991 typedef struct OptixRelocateInputInstanceArray
992 {
995     unsigned int numInstances;
996
1002     CUdeviceptr traversableHandles;
1003
1004 } OptixRelocateInputInstanceArray;
1005
1009 typedef enum OptixBuildInputType
1010 {
1012     OPTIX_BUILD_INPUT_TYPE_TRIANGLES = 0x2141,
1014     OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES = 0x2142,
1016     OPTIX_BUILD_INPUT_TYPE_INSTANCES = 0x2143,
1018     OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS = 0x2144,
1020     OPTIX_BUILD_INPUT_TYPE_CURVES = 0x2145,
1022     OPTIX_BUILD_INPUT_TYPE_SPHERES = 0x2146
1023 } OptixBuildInputType;
1024
1030 typedef struct OptixBuildInput
1031 {
1033     OptixBuildInputType type;
1034
1035     union
1036     {
1038         OptixBuildInputTriangleArray triangleArray;
1040         OptixBuildInputCurveArray curveArray;
1042         OptixBuildInputSphereArray sphereArray;
1044         OptixBuildInputCustomPrimitiveArray customPrimitiveArray;
1046         OptixBuildInputInstanceArray instanceArray;
1047         char pad[1024];
1048     };
1049 } OptixBuildInput;
1050
1054 typedef struct OptixRelocateInput
1055 {
1057     OptixBuildInputType type;
1058
1059     union
1060     {
1062         OptixRelocateInputInstanceArray instanceArray;
1063
1065         OptixRelocateInputTriangleArray triangleArray;
1066

```

```

1068     };
1069 } OptixRelocateInput;
1070
1071 // Some 32-bit tools use this header. This static_assert fails for them because
1072 // the default enum size is 4 bytes, rather than 8, under 32-bit compilers.
1073 // This #ifndef allows them to disable the static assert.
1074
1075 // TODO Define a static assert for C/pre-C++-11
1076 #if defined(__cplusplus) && __cplusplus >= 201103L
1077 static_assert(sizeof(OptixBuildInput) == 8 + 1024, "OptixBuildInput has wrong size");
1078 #endif
1079
1083 typedef enum OptixInstanceFlags
1084 {
1086     OPTIX_INSTANCE_FLAG_NONE = 0,
1087
1091     OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 0,
1092
1095     OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u << 1,
1096
1100     OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT = 1u << 2,
1101
1106     OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u << 3,
1107
1108
1110     OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 4,
1111     OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS = 1u << 5,
1114
1115 } OptixInstanceFlags;
1116
1120 typedef struct OptixInstance
1121 {
1123     float transform[12];
1124
1126     unsigned int instanceId;
1127
1131     unsigned int sbtOffset;
1132
1135     unsigned int visibilityMask;
1136
1138     unsigned int flags;
1139
1141     OptixTraversableHandle traversableHandle;
1142
1144     unsigned int pad[2];
1145 } OptixInstance;
1146
1150 typedef enum OptixBuildFlags
1151 {
1153     OPTIX_BUILD_FLAG_NONE = 0,
1154
1157     OPTIX_BUILD_FLAG_ALLOW_UPDATE = 1u << 0,
1158
1159     OPTIX_BUILD_FLAG_ALLOW_COMPACTION = 1u << 1,
1160
1162     OPTIX_BUILD_FLAG_PREFER_FAST_TRACE = 1u << 2,
1163
1165     OPTIX_BUILD_FLAG_PREFER_FAST_BUILD = 1u << 3,
1166
1176     OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS = 1u << 4,
1177
1180     OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS = 1u << 5,
1181
1185     OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE = 1u << 6,
1186
1190     OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1u << 7,
1191 } OptixBuildFlags;

```

```

1192
1193
1195 typedef enum OptixOpacityMicromapFlags
1196 {
1197     OPTIX_OPACITY_MICROMAP_FLAG_NONE = 0,
1198
1200     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 « 0,
1201
1203     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 « 1,
1204 } OptixOpacityMicromapFlags;
1205
1207 typedef struct OptixOpacityMicromapDesc
1208 {
1210     unsigned int    byteOffset;
1212     unsigned short subdivisionLevel;
1214     unsigned short format;
1215 } OptixOpacityMicromapDesc;
1216
1221 typedef struct OptixOpacityMicromapHistogramEntry
1222 {
1224     unsigned int    count;
1226     unsigned int    subdivisionLevel;
1228     OptixOpacityMicromapFormat format;
1229 } OptixOpacityMicromapHistogramEntry;
1230
1232 typedef struct OptixOpacityMicromapArrayBuildInput
1233 {
1235     unsigned int flags;
1236
1238     CUdeviceptr inputBuffer;
1239
1242     CUdeviceptr perMicromapDescBuffer;
1243
1247     unsigned int perMicromapDescStrideInBytes;
1248
1250     unsigned int numMicromapHistogramEntries;
1253     const OptixOpacityMicromapHistogramEntry* micromapHistogramEntries;
1254 } OptixOpacityMicromapArrayBuildInput;
1255
1257 typedef struct OptixMicromapBufferSizes
1258 {
1259     size_t outputSizeInBytes;
1260     size_t tempSizeInBytes;
1261 } OptixMicromapBufferSizes;
1262
1264 typedef struct OptixMicromapBuffers
1265 {
1267     CUdeviceptr output;
1269     size_t outputSizeInBytes;
1271     CUdeviceptr temp;
1273     size_t tempSizeInBytes;
1274 } OptixMicromapBuffers;
1275
1276
1288 typedef enum OptixBuildOperation
1289 {
1291     OPTIX_BUILD_OPERATION_BUILD = 0x2161,
1293     OPTIX_BUILD_OPERATION_UPDATE = 0x2162,
1294 } OptixBuildOperation;
1295
1299 typedef enum OptixMotionFlags
1300 {
1301     OPTIX_MOTION_FLAG_NONE = 0,
1302     OPTIX_MOTION_FLAG_START_VANISH = 1u « 0,
1303     OPTIX_MOTION_FLAG_END_VANISH = 1u « 1
1304 } OptixMotionFlags;
1305

```

```
1310 typedef struct OptixMotionOptions
1311 {
1314     unsigned short numKeys;
1315
1317     unsigned short flags;
1318
1320     float timeBegin;
1321
1323     float timeEnd;
1324 } OptixMotionOptions;
1325
1329 typedef struct OptixAccelBuildOptions
1330 {
1332     unsigned int buildFlags;
1333
1340     OptixBuildOperation operation;
1341
1343     OptixMotionOptions motionOptions;
1344 } OptixAccelBuildOptions;
1345
1351 typedef struct OptixAccelBufferSizes
1352 {
1355     size_t outputSizeInBytes;
1356
1359     size_t tempSizeInBytes;
1360
1365     size_t tempUpdateSizeInBytes;
1366 } OptixAccelBufferSizes;
1367
1371 typedef enum OptixAccelPropertyType
1372 {
1374     OPTIX_PROPERTY_TYPE_COMPACTED_SIZE = 0x2181,
1375
1377     OPTIX_PROPERTY_TYPE_AABBS = 0x2182,
1378 } OptixAccelPropertyType;
1379
1383 typedef struct OptixAccelEmitDesc
1384 {
1386     CUdeviceptr result;
1387
1389     OptixAccelPropertyType type;
1390 } OptixAccelEmitDesc;
1391
1396 typedef struct OptixRelocationInfo
1397 {
1399     unsigned long long info[4];
1400 } OptixRelocationInfo;
1401
1407 typedef struct OptixStaticTransform
1408 {
1410     OptixTraversableHandle child;
1411
1413     unsigned int pad[2];
1414
1416     float transform[12];
1417
1420     float invTransform[12];
1421 } OptixStaticTransform;
1422
1447 typedef struct OptixMatrixMotionTransform
1448 {
1450     OptixTraversableHandle child;
1451
1454     OptixMotionOptions motionOptions;
1455
1457     unsigned int pad[3];
1458
```

```

1460     float transform[2][12];
1461 } OptixMatrixMotionTransform;
1462
1470 //     [ sx  a  b  pvx ]
1471 // S = [  0  sy  c  pvy ]
1472 //     [  0  0  sz  pvz ]
1481 //     [ 1  0  0  tx ]
1482 // T = [  0  1  0  ty ]
1483 //     [  0  0  1  tz ]
1493 typedef struct OptixSRTData
1494 {
1497     float sx, a, b, pvx, sy, c, pvy, sz, pvz, qx, qy, qz, qw, tx, ty, tz;
1499 } OptixSRTData;
1500
1501 // TODO Define a static assert for C/pre-C++-11
1502 #if defined(__cplusplus) && __cplusplus >= 201103L
1503 static_assert(sizeof(OptixSRTData) == 16 * 4, "OptixSRTData has wrong size");
1504 #endif
1505
1530 typedef struct OptixSRTMotionTransform
1531 {
1533     OptixTraversableHandle child;
1534
1537     OptixMotionOptions motionOptions;
1538
1540     unsigned int pad[3];
1541
1543     OptixSRTData srtData[2];
1544 } OptixSRTMotionTransform;
1545
1546 // TODO Define a static assert for C/pre-C++-11
1547 #if defined(__cplusplus) && __cplusplus >= 201103L
1548 static_assert(sizeof(OptixSRTMotionTransform) == 8 + 12 + 12 + 2 * 16 * 4, "OptixSRTMotionTransform has
wrong size");
1549 #endif
1550
1554 typedef enum OptixTraversableType
1555 {
1557     OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM = 0x21C1,
1559     OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM = 0x21C2,
1561     OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM = 0x21C3,
1562 } OptixTraversableType;
1563
1567 typedef enum OptixPixelFormat
1568 {
1569     OPTIX_PIXEL_FORMAT_HALF1 = 0x220a,
1570     OPTIX_PIXEL_FORMAT_HALF2 = 0x2207,
1571     OPTIX_PIXEL_FORMAT_HALF3 = 0x2201,
1572     OPTIX_PIXEL_FORMAT_HALF4 = 0x2202,
1573     OPTIX_PIXEL_FORMAT_FLOAT1 = 0x220b,
1574     OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208,
1575     OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203,
1576     OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204,
1577     OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205,
1578     OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206,
1579     OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209
1580 } OptixPixelFormat;
1581
1585 typedef struct OptixImage2D
1586 {
1588     CUdeviceptr data;
1590     unsigned int width;
1592     unsigned int height;
1594     unsigned int rowStrideInBytes;
1599     unsigned int pixelStrideInBytes;
1601     OptixPixelFormat format;
1602 } OptixImage2D;

```

```

1603
1607 typedef enum OptixDenoiserModelKind
1608 {
1610     OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322,
1611
1613     OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323,
1614
1616     OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324,
1617
1619     OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325,
1620
1622     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326,
1623
1625     OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327,
1626
1629     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328
1630 } OptixDenoiserModelKind;
1631
1635 typedef enum OptixDenoiserAlphaMode
1636 {
1638     OPTIX_DENOISER_ALPHA_MODE_COPY = 0,
1639
1641     OPTIX_DENOISER_ALPHA_MODE_DENOISE = 1
1642 } OptixDenoiserAlphaMode;
1643
1647 typedef struct OptixDenoiserOptions
1648 {
1649     // if nonzero, albedo image must be given in OptixDenoiserGuideLayer
1650     unsigned int guideAlbedo;
1651
1652     // if nonzero, normal image must be given in OptixDenoiserGuideLayer
1653     unsigned int guideNormal;
1654
1656     OptixDenoiserAlphaMode denoiseAlpha;
1657 } OptixDenoiserOptions;
1658
1662 typedef struct OptixDenoiserGuideLayer
1663 {
1664     // image with three components: R, G, B.
1665     OptixImage2D albedo;
1666
1667     // image with two or three components: X, Y, Z.
1668     // (X, Y) camera space for OPTIX_DENOISER_MODEL_KIND_LDR, OPTIX_DENOISER_MODEL_KIND_HDR models.
1669     // (X, Y, Z) world space, all other models.
1670     OptixImage2D normal;
1671
1672     // image with two components: X, Y.
1673     // pixel movement from previous to current frame for each pixel in screen space.
1674     OptixImage2D flow;
1675
1676     // Internal images used in temporal AOV denoising modes,
1677     // pixel format OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER.
1678     OptixImage2D previousOutputInternalGuideLayer;
1679     OptixImage2D outputInternalGuideLayer;
1680
1681     // image with a single component value that specifies how trustworthy the flow vector at x,y
1682     // position in OptixDenoiserGuideLayer::flow is. Range 0..1 (low->high trustworthiness).
1683     // Ignored if data pointer in the image is zero.
1684     OptixImage2D flowTrustworthiness;
1685
1686 } OptixDenoiserGuideLayer;
1687
1690 typedef enum OptixDenoiserAOVType
1691 {
1693     OPTIX_DENOISER_AOV_TYPE_NONE = 0,
1694

```

```

1695     OPTIX_DENOISER_AOV_TYPE_BEAUTY      = 0x7000,
1696     OPTIX_DENOISER_AOV_TYPE_SPECULAR   = 0x7001,
1697     OPTIX_DENOISER_AOV_TYPE_REFLECTION = 0x7002,
1698     OPTIX_DENOISER_AOV_TYPE_REFRACTION = 0x7003,
1699     OPTIX_DENOISER_AOV_TYPE_DIFFUSE    = 0x7004
1700
1701 } OptixDenoiserAOVType;
1702
1706 typedef struct OptixDenoiserLayer
1707 {
1708     // input image (beauty or AOV)
1709     OptixImage2D input;
1710
1711     // denoised output image from previous frame if temporal model kind selected
1712     OptixImage2D previousOutput;
1713
1714     // denoised output for given input
1715     OptixImage2D output;
1716
1717     // Type of AOV, used in temporal AOV modes as a hint to improve image quality.
1718     OptixDenoiserAOVType type;
1719 } OptixDenoiserLayer;
1720
1726
1727 typedef struct OptixDenoiserParams
1728 {
1733     CUdeviceptr  hdrIntensity;
1734
1739     float        blendFactor;
1740
1746     CUdeviceptr  hdrAverageColor;
1747
1752     unsigned int temporalModeUsePreviousLayers;
1753 } OptixDenoiserParams;
1754
1758 typedef struct OptixDenoiserSizes
1759 {
1761     size_t stateSizeInBytes;
1762
1765     size_t withOverlapScratchSizeInBytes;
1766
1769     size_t withoutOverlapScratchSizeInBytes;
1770
1772     unsigned int overlapWindowSizeInPixels;
1773
1776     size_t computeAverageColorSizeInBytes;
1777
1780     size_t computeIntensitySizeInBytes;
1781
1783     size_t internalGuideLayerPixelSizeInBytes;
1784 } OptixDenoiserSizes;
1785
1790 typedef enum OptixRayFlags
1791 {
1793     OPTIX_RAY_FLAG_NONE = 0u,
1794
1799     OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u << 0,
1800
1805     OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u << 1,
1806
1809     OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u << 2,
1810
1812     OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u << 3,
1813
1818     OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u << 4,
1819
1824     OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u << 5,

```

```

1825
1831     OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u << 6,
1832
1838     OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u << 7,
1839
1841     OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 10,
1842 } OptixRayFlags;
1843
1849 typedef enum OptixTransformType
1850 {
1851     OPTIX_TRANSFORM_TYPE_NONE = 0,
1852     OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM = 1,
1853     OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM = 2,
1854     OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM = 3,
1855     OPTIX_TRANSFORM_TYPE_INSTANCE = 4,
1856 } OptixTransformType;
1857
1860 typedef enum OptixTraversableGraphFlags
1861 {
1864     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0,
1865
1869     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u << 0,
1870
1875     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u << 1,
1876 } OptixTraversableGraphFlags;
1877
1881 typedef enum OptixCompileOptimizationLevel
1882 {
1884     OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0,
1886     OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340,
1888     OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341,
1890     OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342,
1892     OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343,
1893 } OptixCompileOptimizationLevel;
1894
1898 typedef enum OptixCompileDebugLevel
1899 {
1901     OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT = 0,
1903     OPTIX_COMPILE_DEBUG_LEVEL_NONE = 0x2350,
1906     OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL = 0x2351,
1908     OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353,
1910     OPTIX_COMPILE_DEBUG_LEVEL_FULL = 0x2352,
1911 } OptixCompileDebugLevel;
1912
1916 typedef enum OptixModuleCompileState
1917 {
1919     OPTIX_MODULE_COMPILE_STATE_NOT_STARTED = 0x2360,
1920
1922     OPTIX_MODULE_COMPILE_STATE_STARTED = 0x2361,
1923
1925     OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE = 0x2362,
1926
1928     OPTIX_MODULE_COMPILE_STATE_FAILED = 0x2363,
1929
1931     OPTIX_MODULE_COMPILE_STATE_COMPLETED = 0x2364,
1932 } OptixModuleCompileState;
1933
1934
1935
1968 typedef struct OptixModuleCompileBoundValueEntry {
1969     size_t pipelineParamOffsetInBytes;
1970     size_t sizeInBytes;
1971     const void* boundValuePtr;
1972     const char* annotation; // optional string to display, set to 0 if unused. If unused,
1973                             // OptiX will report the annotation as "No annotation"
1974 } OptixModuleCompileBoundValueEntry;
1975

```

```

1977 typedef enum OptixPayloadTypeID {
1978     OPTIX_PAYLOAD_TYPE_DEFAULT = 0,
1979     OPTIX_PAYLOAD_TYPE_ID_0 = (1 << 0u),
1980     OPTIX_PAYLOAD_TYPE_ID_1 = (1 << 1u),
1981     OPTIX_PAYLOAD_TYPE_ID_2 = (1 << 2u),
1982     OPTIX_PAYLOAD_TYPE_ID_3 = (1 << 3u),
1983     OPTIX_PAYLOAD_TYPE_ID_4 = (1 << 4u),
1984     OPTIX_PAYLOAD_TYPE_ID_5 = (1 << 5u),
1985     OPTIX_PAYLOAD_TYPE_ID_6 = (1 << 6u),
1986     OPTIX_PAYLOAD_TYPE_ID_7 = (1 << 7u)
1987 } OptixPayloadTypeID;
1988
2002 typedef enum OptixPayloadSemantics
2003 {
2004     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE           = 0,
2005     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ         = 1u << 0,
2006     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE        = 2u << 0,
2007     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE   = 3u << 0,
2008
2009     OPTIX_PAYLOAD_SEMANTICS_CH_NONE                   = 0,
2010     OPTIX_PAYLOAD_SEMANTICS_CH_READ                   = 1u << 2,
2011     OPTIX_PAYLOAD_SEMANTICS_CH_WRITE                  = 2u << 2,
2012     OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE            = 3u << 2,
2013
2014     OPTIX_PAYLOAD_SEMANTICS_MS_NONE                   = 0,
2015     OPTIX_PAYLOAD_SEMANTICS_MS_READ                   = 1u << 4,
2016     OPTIX_PAYLOAD_SEMANTICS_MS_WRITE                  = 2u << 4,
2017     OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE            = 3u << 4,
2018
2019     OPTIX_PAYLOAD_SEMANTICS_AH_NONE                   = 0,
2020     OPTIX_PAYLOAD_SEMANTICS_AH_READ                   = 1u << 6,
2021     OPTIX_PAYLOAD_SEMANTICS_AH_WRITE                  = 2u << 6,
2022     OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE            = 3u << 6,
2023
2024     OPTIX_PAYLOAD_SEMANTICS_IS_NONE                   = 0,
2025     OPTIX_PAYLOAD_SEMANTICS_IS_READ                   = 1u << 8,
2026     OPTIX_PAYLOAD_SEMANTICS_IS_WRITE                  = 2u << 8,
2027     OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE            = 3u << 8,
2028 } OptixPayloadSemantics;
2029
2031 typedef struct OptixPayloadType
2032 {
2033     unsigned int numPayloadValues;
2034
2035     const unsigned int *payloadSemantics;
2036 } OptixPayloadType;
2037
2043 typedef struct OptixModuleCompileOptions
2044 {
2045     int maxRegisterCount;
2046
2047     OptixCompileOptimizationLevel optLevel;
2048
2049     OptixCompileDebugLevel debugLevel;
2050
2051     const OptixModuleCompileBoundValueEntry* boundValues;
2052
2053     unsigned int numBoundValues;
2054
2055     unsigned int numPayloadTypes;
2056
2057     const OptixPayloadType* payloadTypes;
2058 } OptixModuleCompileOptions;
2059
2071 typedef enum OptixProgramGroupKind
2072 {

```

```

2075     OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421,
2076
2079     OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422,
2080
2083     OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423,
2084
2087     OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424,
2088
2091     OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425
2092 } OptixProgramGroupKind;
2093
2095 typedef enum OptixProgramGroupFlags
2096 {
2098     OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0
2099 } OptixProgramGroupFlags;
2100
2107 typedef struct OptixProgramGroupSingleModule
2108 {
2110     OptixModule module;
2112     const char* entryFunctionName;
2113 } OptixProgramGroupSingleModule;
2114
2120 typedef struct OptixProgramGroupHitgroup
2121 {
2123     OptixModule moduleCH;
2125     const char* entryFunctionNameCH;
2127     OptixModule moduleAH;
2129     const char* entryFunctionNameAH;
2131     OptixModule moduleIS;
2133     const char* entryFunctionNameIS;
2134 } OptixProgramGroupHitgroup;
2135
2141 typedef struct OptixProgramGroupCallables
2142 {
2144     OptixModule moduleDC;
2146     const char* entryFunctionNameDC;
2148     OptixModule moduleCC;
2150     const char* entryFunctionNameCC;
2151 } OptixProgramGroupCallables;
2152
2154 typedef struct OptixProgramGroupDesc
2155 {
2157     OptixProgramGroupKind kind;
2158
2160     unsigned int flags;
2161
2162     union
2163     {
2165         OptixProgramGroupSingleModule raygen;
2167         OptixProgramGroupSingleModule miss;
2169         OptixProgramGroupSingleModule exception;
2171         OptixProgramGroupCallables callables;
2173         OptixProgramGroupHitgroup hitgroup;
2174     };
2175 } OptixProgramGroupDesc;
2176
2180 typedef struct OptixProgramGroupOptions
2181 {
2194     const OptixPayloadType* payloadType;
2195 } OptixProgramGroupOptions;
2196
2198 typedef enum OptixExceptionCodes
2199 {
2202     OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1,
2203
2206     OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2,
2207

```

```
2208
2209 } OptixExceptionCodes;
2210
2214 typedef enum OptixExceptionFlags
2215 {
2217     OPTIX_EXCEPTION_FLAG_NONE = 0,
2218
2225     OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u << 0,
2226
2233     OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u << 1,
2234
2237     OPTIX_EXCEPTION_FLAG_USER = 1u << 2,
2238
2239 } OptixExceptionFlags;
2240
2246 typedef struct OptixPipelineCompileOptions
2247 {
2249     int usesMotionBlur;
2250
2252     unsigned int traversableGraphFlags;
2253
2256     int numPayloadValues;
2257
2260     int numAttributeValues;
2261
2263     unsigned int exceptionFlags;
2264
2268     const char* pipelineLaunchParamsVariableName;
2269
2272     unsigned int usesPrimitiveTypeFlags;
2273
2275     int allowOpacityMicromaps;
2276 } OptixPipelineCompileOptions;
2277
2281 typedef struct OptixPipelineLinkOptions
2282 {
2285     unsigned int maxTraceDepth;
2286
2287 } OptixPipelineLinkOptions;
2288
2292 typedef struct OptixShaderBindingTable
2293 {
2296     CUdeviceptr raygenRecord;
2297
2300     CUdeviceptr exceptionRecord;
2301
2305     CUdeviceptr missRecordBase;
2306     unsigned int missRecordStrideInBytes;
2307     unsigned int missRecordCount;
2309
2313     CUdeviceptr hitgroupRecordBase;
2314     unsigned int hitgroupRecordStrideInBytes;
2315     unsigned int hitgroupRecordCount;
2317
2322     CUdeviceptr callablesRecordBase;
2323     unsigned int callablesRecordStrideInBytes;
2324     unsigned int callablesRecordCount;
2326
2327 } OptixShaderBindingTable;
2328
2332 typedef struct OptixStackSizes
2333 {
2335     unsigned int cssRG;
2337     unsigned int cssMS;
2339     unsigned int cssCH;
2341     unsigned int cssAH;
2343     unsigned int cssIS;
```

```
2345     unsigned int cssCC;
2347     unsigned int dssDC;
2348
2349 } OptixStackSizes;
2350
2352 typedef enum OptixQueryFunctionTableOptions
2353 {
2355     OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0
2356
2357 } OptixQueryFunctionTableOptions;
2358
2360 typedef OptixResult(OptixQueryFunctionTable_t)(int         abiId,
2361                                                unsigned int numOptions,
2362                                                OptixQueryFunctionTableOptions* /*optionKeys*/,
2363                                                const void** /*optionValues*/,
2364                                                void* functionTable,
2365                                                size_t sizeOfTable);
2366
2371 typedef struct OptixBuiltinISOptions
2372 {
2373     OptixPrimitiveType    builtinISModuleType;
2375     int                   usesMotionBlur;
2377     unsigned int          buildFlags;
2379     unsigned int          curveEndcapFlags;
2380 } OptixBuiltinISOptions;
2381
2382
2383 // end group optix_types
2385
2386 #endif // OPTIX_OPTIX_TYPES_H
```

8.27 main.dox File Reference