



NVIDIA OptiX 9.1

API Reference Manual

18 November 2025
Version 9.1



Table of Contents

1	NVIDIA OptiX 9.1 API	1
2	Module Index	1
2.1	Modules	1
3	Class Index	1
3.1	Class List	1
4	File Index	5
4.1	File List	5
5	Module Documentation	6
5.1	Device API	6
5.2	Cooperative Vector	70
5.3	Function Table	78
5.4	Host API	79
5.5	Error handling	79
5.6	Device context	80
5.7	Pipelines	85
5.8	Modules	88
5.9	Tasks	91
5.10	Program groups	94
5.11	Launches	95
5.12	Acceleration structures	96
5.13	Cooperative Vector	105
5.14	Denoiser	107
5.15	Utilities	113
5.16	Types	120
6	Namespace Documentation	176
6.1	optix_impl Namespace Reference	176
6.2	optix_internal Namespace Reference	180
7	Class Documentation	180
7.1	OptixAabb Struct Reference	180
7.2	OptixAccelBufferSizes Struct Reference	181
7.3	OptixAccelBuildOptions Struct Reference	182
7.4	OptixAccelEmitDesc Struct Reference	182
7.5	OptixBuildInput Struct Reference	183
7.6	OptixBuildInputCurveArray Struct Reference	184
7.7	OptixBuildInputCustomPrimitiveArray Struct Reference	186
7.8	OptixBuildInputInstanceArray Struct Reference	188
7.9	OptixBuildInputOpacityMicromap Struct Reference	188
7.10	OptixBuildInputSphereArray Struct Reference	190
7.11	OptixBuildInputTriangleArray Struct Reference	192
7.12	OptixBuiltinISOOptions Struct Reference	195
7.13	OptixClusterAccelBuildInput Struct Reference	195
7.14	OptixClusterAccelBuildInputClusters Struct Reference	196
7.15	OptixClusterAccelBuildInputClustersArgs Struct Reference	197
7.16	OptixClusterAccelBuildInputGrids Struct Reference	197
7.17	OptixClusterAccelBuildInputGridsArgs Struct Reference	198
7.18	OptixClusterAccelBuildInputTemplatesArgs Struct Reference	199
7.19	OptixClusterAccelBuildInputTriangles Struct Reference	200

7.20	OptixClusterAccelBuildInputTrianglesArgs Struct Reference	202
7.21	OptixClusterAccelBuildModeDesc Struct Reference	204
7.22	OptixClusterAccelBuildModeDescExplicitDest Struct Reference	205
7.23	OptixClusterAccelBuildModeDescGetSize Struct Reference	206
7.24	OptixClusterAccelBuildModeDescImplicitDest Struct Reference	207
7.25	OptixClusterAccelPrimitiveInfo Struct Reference	208
7.26	OptixCoopVec< T, N > Class Template Reference	208
7.27	OptixCoopVecMatrixDescription Struct Reference	210
7.28	OptixDenoiserGuideLayer Struct Reference	211
7.29	OptixDenoiserLayer Struct Reference	212
7.30	OptixDenoiserOptions Struct Reference	212
7.31	OptixDenoiserParams Struct Reference	213
7.32	OptixDenoiserSizes Struct Reference	214
7.33	OptixDeviceContextOptions Struct Reference	215
7.34	OptixFunctionTable Struct Reference	216
7.35	OptixImage2D Struct Reference	227
7.36	OptixIncomingHitObject Struct Reference	228
7.37	OptixInstance Struct Reference	229
7.38	OptixMatrixMotionTransform Struct Reference	230
7.39	OptixMicromapBuffers Struct Reference	231
7.40	OptixMicromapBufferSizes Struct Reference	232
7.41	OptixModuleCompileBoundValueEntry Struct Reference	232
7.42	OptixModuleCompileOptions Struct Reference	233
7.43	OptixMotionOptions Struct Reference	234
7.44	OptixNetworkDescription Struct Reference	235
7.45	OptixOpacityMicromapArrayBuildInput Struct Reference	236
7.46	OptixOpacityMicromapDesc Struct Reference	237
7.47	OptixOpacityMicromapHistogramEntry Struct Reference	237
7.48	OptixOpacityMicromapUsageCount Struct Reference	238
7.49	OptixOutgoingHitObject Struct Reference	238
7.50	OptixPayloadType Struct Reference	239
7.51	OptixPipelineCompileOptions Struct Reference	239
7.52	OptixPipelineLinkOptions Struct Reference	241
7.53	OptixProgramGroupCallables Struct Reference	242
7.54	OptixProgramGroupDesc Struct Reference	243
7.55	OptixProgramGroupHitgroup Struct Reference	244
7.56	OptixProgramGroupOptions Struct Reference	245
7.57	OptixProgramGroupSingleModule Struct Reference	245
7.58	OptixRelocateInput Struct Reference	246
7.59	OptixRelocateInputInstanceArray Struct Reference	247
7.60	OptixRelocateInputOpacityMicromap Struct Reference	247
7.61	OptixRelocateInputTriangleArray Struct Reference	248
7.62	OptixRelocationInfo Struct Reference	248
7.63	OptixShaderBindingTable Struct Reference	248
7.64	OptixSRTData Struct Reference	250
7.65	OptixSRTMotionTransform Struct Reference	252
7.66	OptixStackSizes Struct Reference	253
7.67	OptixStaticTransform Struct Reference	255
7.68	OptixTraverseData Struct Reference	255
7.69	OptixUtilDenoiserImageTile Struct Reference	256
7.70	optix_internal::TypePack<... > Struct Template Reference	256

8 File Documentation

256

8.1	<code>optix_device_impl.h</code> File Reference	256
8.2	<code>optix_device_impl.h</code>	297
8.3	<code>optix_device_impl_transformations.h</code> File Reference	341
8.4	<code>optix_device_impl_transformations.h</code>	342
8.5	<code>optix_micromap_impl.h</code> File Reference	349
8.6	<code>optix_micromap_impl.h</code>	350
8.7	<code>optix.h</code> File Reference	352
8.8	<code>optix.h</code>	353
8.9	<code>optix_denoiser_tiling.h</code> File Reference	353
8.10	<code>optix_denoiser_tiling.h</code>	354
8.11	<code>optix_device.h</code> File Reference	359
8.12	<code>optix_device.h</code>	369
8.13	<code>optix_function_table.h</code> File Reference	383
8.14	<code>optix_function_table.h</code>	383
8.15	<code>optix_function_table_definition.h</code> File Reference	389
8.16	<code>optix_function_table_definition.h</code>	389
8.17	<code>optix_host.h</code> File Reference	390
8.18	<code>optix_host.h</code>	393
8.19	<code>optix_micromap.h</code> File Reference	399
8.20	<code>optix_micromap.h</code>	400
8.21	<code>optix_stack_size.h</code> File Reference	401
8.22	<code>optix_stack_size.h</code>	402
8.23	<code>optix_stubs.h</code> File Reference	406
8.24	<code>optix_stubs.h</code>	406
8.25	<code>optix_types.h</code> File Reference	420
8.26	<code>optix_types.h</code>	431
8.27	<code>main.dox</code> File Reference	454

1 NVIDIA OptiX 9.1 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	6
Cooperative Vector	70
Function Table	78
Host API	79
Error handling	79
Device context	80
Pipelines	85
Modules	88
Tasks	91
Program groups	94
Launches	95
Acceleration structures	96
Cooperative Vector	105
Denoiser	107
Utilities	113
Types	120

3 Class Index

3.1 Class List

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

<code>OptixAabb</code>	
AABB inputs	180
<code>OptixAccelBufferSizes</code>	
Struct for querying builder allocation requirements	181
<code>OptixAccelBuildOptions</code>	
Build options for acceleration structures	182
<code>OptixAccelEmitDesc</code>	
Specifies a type and output destination for emitted post-build properties	182
<code>OptixBuildInput</code>	
Build inputs	183
<code>OptixBuildInputCurveArray</code>	
Curve inputs	184

<code>OptixBuildInputCustomPrimitiveArray</code>	
Custom primitive inputs	186
<code>OptixBuildInputInstanceArray</code>	
Instance and instance pointer inputs	188
<code>OptixBuildInputOpacityMicromap</code>	188
<code>OptixBuildInputSphereArray</code>	
Sphere inputs	190
<code>OptixBuildInputTriangleArray</code>	
Triangle inputs	192
<code>OptixBuiltinISOptions</code>	
Specifies the options for retrieving an intersection program for a built-in primitive type. The primitive type must not be <code>OPTIX_PRIMITIVE_TYPE_CUSTOM</code>	195
<code>OptixClusterAccelBuildInput</code>	195
<code>OptixClusterAccelBuildInputClusters</code>	196
<code>OptixClusterAccelBuildInputClustersArgs</code>	
Device data, args provided for <code>OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS</code> builds	197
<code>OptixClusterAccelBuildInputGrids</code>	197
<code>OptixClusterAccelBuildInputGridsArgs</code>	
Device data, args provided for <code>OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS</code> builds	198
<code>OptixClusterAccelBuildInputTemplatesArgs</code>	
Device data, args provided for <code>OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES</code> builds	199
<code>OptixClusterAccelBuildInputTriangles</code>	200
<code>OptixClusterAccelBuildInputTrianglesArgs</code>	
Device data, args provided for <code>OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES</code> builds and <code>OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES</code> builds	202
<code>OptixClusterAccelBuildModeDesc</code>	204
<code>OptixClusterAccelBuildModeDescExplicitDest</code>	205
<code>OptixClusterAccelBuildModeDescGetSize</code>	206
<code>OptixClusterAccelBuildModeDescImplicitDest</code>	207
<code>OptixClusterAccelPrimitiveInfo</code>	208
<code>OptixCoopVec< T, N ></code>	
The API does not require the use of this class specifically, but it must define a certain interface as spelled out by the public members of the class. Note that not all types of T are supported. Only 8 and 32 bit signed and unsigned integral types along with 16 and 32 bit floating point values	208
<code>OptixCoopVecMatrixDescription</code>	
Each matrix's offset from the base address is expressed with <code>offsetInBytes</code> . This allows for non-uniform matrices to be tightly packed	210
<code>OptixDenoiserGuideLayer</code>	
Guide layer for the denoiser	211

<code>OptixDenoiserLayer</code>	
Input/Output layers for the denoiser	212
<code>OptixDenoiserOptions</code>	
Options used by the denoiser	212
<code>OptixDenoiserParams</code>	
Various parameters used by the denoiser	213
<code>OptixDenoiserSizes</code>	
Various sizes related to the denoiser	214
<code>OptixDeviceContextOptions</code>	
Parameters used for <code>optixDeviceContextCreate()</code>	215
<code>OptixFunctionTable</code>	
The function table containing all API functions	216
<code>OptixImage2D</code>	
Image descriptor used by the denoiser	227
<code>OptixIncomingHitObject</code>	228
<code>OptixInstance</code>	
Instances	229
<code>OptixMatrixMotionTransform</code>	
Represents a matrix motion transformation	230
<code>OptixMicromapBuffers</code>	
Buffer inputs for opacity micromap array builds	231
<code>OptixMicromapBufferSizes</code>	
Conservative memory requirements for building a opacity micromap array	232
<code>OptixModuleCompileBoundValueEntry</code>	
Struct for specifying specializations for pipelineParams as specified in <code>OptixPipelineCompileOptions::pipelineLaunchParamsVariableName</code>	232
<code>OptixModuleCompileOptions</code>	
Compilation options for module	233
<code>OptixMotionOptions</code>	
Motion options	234
<code>OptixNetworkDescription</code>	235
<code>OptixOpacityMicromapArrayBuildInput</code>	
Inputs to opacity micromap array construction	236
<code>OptixOpacityMicromapDesc</code>	
Opacity micromap descriptor	237
<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	237

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	238
OptixOutgoingHitObject	238
OptixPayloadType	
Specifies a single payload type	239
OptixPipelineCompileOptions	
Compilation options for all modules of a pipeline	239
OptixPipelineLinkOptions	
Link options for a pipeline	241
OptixProgramGroupCallables	
Program group representing callables	242
OptixProgramGroupDesc	
Descriptor for program groups	243
OptixProgramGroupHitgroup	
Program group representing the hitgroup	244
OptixProgramGroupOptions	
Program group options	245
OptixProgramGroupSingleModule	
Program group representing a single module	245
OptixRelocateInput	
Relocation inputs	246
OptixRelocateInputInstanceArray	
Instance and instance pointer inputs	247
OptixRelocateInputOpacityMicromap	247
OptixRelocateInputTriangleArray	
Triangle inputs	248
OptixRelocationInfo	
Used to store information related to relocation of optix data structures	248
OptixShaderBindingTable	
Describes the shader binding table (SBT)	248
OptixSRTData	
Represents an SRT transformation	250
OptixSRTMotionTransform	
Represents an SRT motion transformation	252
OptixStackSizes	
Describes the stack size requirements of a program group	253
OptixStaticTransform	
Static transform	255

<code>OptixTraverseData</code>	
Hit Object Struct to store the data collected in a hit object during traversal in an internal format using <code>optixHitObjectGetTraverseData()</code> . The hit object can be reconstructed using that data at a later point with <code>optixMakeHitObjectWithTraverseData()</code>	255
<code>OptixUtilDenoiserImageTile</code>	
Tile definition	256
<code>optix_internal::TypePack<... ></code>	256

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	256
<code>optix_device_impl_transformations.h</code>	
OptiX public API	341
<code>optix_micromap_impl.h</code>	
OptiX micromap helper functions	349
<code>optix.h</code>	
OptiX public API header	352
<code>optix_denoiser_tiling.h</code>	
OptiX public API header	353
<code>optix_device.h</code>	
OptiX public API header	359
<code>optix_function_table.h</code>	
OptiX public API header	383
<code>optix_function_table_definition.h</code>	
OptiX public API header	389
<code>optix_host.h</code>	
OptiX public API header	390
<code>optix_micromap.h</code>	
OptiX micromap helper functions	399
<code>optix_stack_size.h</code>	
OptiX public API header	401
<code>optix_stubs.h</code>	
OptiX public API header	406
<code>optix_types.h</code>	
OptiX public API header	420

5 Module Documentation

5.1 Device API

Modules

- [Cooperative Vector](#)

Classes

- struct [OptixIncomingHitObject](#)
- struct [OptixOutgoingHitObject](#)

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)`
- `static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float rayTime, unsigned int rayFlags, OptixTraverseData traverseData, const OptixTraversableHandle *transforms, unsigned int numTransforms)`
- `static __forceinline__ __device__ void optixMakeMissHitObject (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int rayFlags)`
- `static __forceinline__ __device__ void optixMakeNopHitObject ()`
- `static __forceinline__ __device__ void optixHitObjectGetTraverseData (OptixTraverseData *data)`
- `static __forceinline__ __device__ bool optixHitObjectIsHit ()`
- `static __forceinline__ __device__ bool optixHitObjectIsMiss ()`
- `static __forceinline__ __device__ bool optixHitObjectIsNop ()`

- [illegible]

- 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__ unsigned int optixGetRemainingTraceDepth ()
- 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 optixHitObjectGetRayFlags ()
- 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 optixGetTriangleVertexDataFromHandle ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static __forceinline__ __device__ void optixGetTriangleVertexData (float3 data[3])
- static __forceinline__ __device__ void optixHitObjectGetTriangleVertexData (float3 data[3])
- static __forceinline__ __device__ void optixGetLinearCurveVertexData ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void optixGetLinearCurveVertexDataFromHandle ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])

- `static __forceinline__ __device__ void optixGetLinearCurveVertexData (float4 data[2])`
- `static __forceinline__ __device__ void optixHitObjectGetLinearCurveVertexData (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 optixGetQuadraticBSplineVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])`
- `static __forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])`
- `static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (float4 data[3])`
- `static __forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexData (float4 data[3])`
- `static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineVertexData (float4 data[3])`
- `static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineRocapsVertexData (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 optixGetCubicBSplineVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])`
- `static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])`
- `static __forceinline__ __device__ void optixGetCubicBSplineVertexData (float4 data[4])`
- `static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexData (float4 data[4])`
- `static __forceinline__ __device__ void optixHitObjectGetCubicBSplineVertexData (float4 data[4])`
- `static __forceinline__ __device__ void optixHitObjectGetCubicBSplineRocapsVertexData (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 optixGetCatmullRomVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])`
- `static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])`
- `static __forceinline__ __device__ void optixGetCatmullRomVertexData (float4 data[4])`
- `static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexData (float4 data[4])`
- `static __forceinline__ __device__ void optixHitObjectGetCatmullRomVertexData (float4 data[4])`
- `static __forceinline__ __device__ void optixHitObjectGetCatmullRomRocapsVertexData (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 optixGetCubicBezierVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])`
- `static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])`

- static `__forceinline__ __device__ void optixGetCubicBezierVertexData (float4 data[4])`
- static `__forceinline__ __device__ void optixGetCubicBezierRocapsVertexData (float4 data[4])`
- static `__forceinline__ __device__ void optixHitObjectGetCubicBezierVertexData (float4 data[4])`
- static `__forceinline__ __device__ void optixHitObjectGetCubicBezierRocapsVertexData (float4 data[4])`
- static `__forceinline__ __device__ void optixGetRibbonVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])`
- static `__forceinline__ __device__ void optixGetRibbonVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])`
- static `__forceinline__ __device__ void optixGetRibbonVertexData (float4 data[3])`
- static `__forceinline__ __device__ void optixHitObjectGetRibbonVertexData (float4 data[3])`
- static `__forceinline__ __device__ float3 optixGetRibbonNormal (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)`
- static `__forceinline__ __device__ float3 optixGetRibbonNormalFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)`
- static `__forceinline__ __device__ float3 optixGetRibbonNormal (float2 ribbonParameters)`
- static `__forceinline__ __device__ float3 optixHitObjectGetRibbonNormal (float2 ribbonParameters)`
- static `__forceinline__ __device__ void optixGetSphereData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])`
- static `__forceinline__ __device__ void optixGetSphereDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])`
- static `__forceinline__ __device__ void optixGetSphereData (float4 data[1])`
- static `__forceinline__ __device__ void optixHitObjectGetSphereData (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__ void optixHitObjectGetWorldToObjectTransformMatrix (float m[12])`
- static `__forceinline__ __device__ void optixHitObjectGetObjectToWorldTransformMatrix (float m[12])`

- `static __forceinline__ __device__ float3 optixHitObjectTransformPointFromWorldToObjectSpace (float3 point)`
- `static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromWorldToObjectSpace (float3 vec)`
- `static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromWorldToObjectSpace (float3 normal)`
- `static __forceinline__ __device__ float3 optixHitObjectTransformPointFromObjectToWorldSpace (float3 point)`
- `static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromObjectToWorldSpace (float3 vec)`
- `static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromObjectToWorldSpace (float3 normal)`
- `template<typename HitState > static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (const HitState &hs, float m[12])`
- `template<typename HitState > static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (const HitState &hs, float m[12])`
- `template<typename HitState > static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (const HitState &hs, float3 point)`
- `template<typename HitState > static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (const HitState &hs, float3 vec)`
- `template<typename HitState > static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (const HitState &hs, float3 normal)`
- `template<typename HitState > static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (const HitState &hs, float3 point)`
- `template<typename HitState > static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (const HitState &hs, float3 vec)`
- `template<typename HitState > static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (const HitState &hs, 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 [optixGetClusterId](#) ()
- static __forceinline__ __device__ unsigned int [optixHitObjectGetClusterId](#) ()
- 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__ float2 optixGetTriangleBarycentrics ()
- static __forceinline__ __device__ float2 optixHitObjectGetTriangleBarycentrics ()
- static __forceinline__ __device__ float optixGetCurveParameter ()
- static __forceinline__ __device__ float optixHitObjectGetCurveParameter ()
- static __forceinline__ __device__ float2 optixGetRibbonParameters ()
- static __forceinline__ __device__ float2 optixHitObjectGetRibbonParameters ()
- 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 OptixShaderBindingTable::callablesRecordBase .
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 in 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 OptixShaderBindingTable::callablesRecordBase .
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 optixGetCatmullRomRocapsVertexData()

```
static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexData (
    float4 data[4] ) [static]
```

5.1.2.12 optixGetCatmullRomRocapsVertexDataFromHandle()

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

5.1.2.13 optixGetCatmullRomVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetCatmullRomVertexData (
    float4 data[4] ) [static]
```

Returns the object space curve control vertex data of a CatmullRom spline curve in a Geometry Acceleration Structure (GAS) at a given motion time.

data[i] = {x,y,z,w} with {x,y,z} the position and w the radius of control vertex i.

Available in AH, CH

5.1.2.14 optixGetCatmullRomVertexData() [2/2]

```
static __forceinline__ __device__ void optixGetCatmullRomVertexData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
```

```
float4 data[4] ) [static]
```

Deprecated. Call either `optixGetCatmullRomVertexData(float4 data[4])` for current hit data, or `optixGetCatmullRomVertexDataFromHandle()` for random access sphere data.

Returns 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

5.1.2.15 `optixGetCatmullRomVertexDataFromHandle()`

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

Returns 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

5.1.2.16 `optixGetClusterId()`

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

Returns the user-provided cluster ID of the intersected CLAS of a hit.

Returns `OPTIX_CLUSTER_ID_INVALID` if the closest (or current) intersection is not a cluster.

see also `OptixPipelineCompileOptions::allowClusteredGeometry`

Available in AH, CH

5.1.2.17 `optixGetCubicBezierRocapsVertexData()`

```
static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexData (
    float4 data[4] ) [static]
```

5.1.2.18 optixGetCubicBezierRocapsVertexDataFromHandle()

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

5.1.2.19 optixGetCubicBezierVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetCubicBezierVertexData (
    float4 data[4] ) [static]
```

Returns the object space curve control vertex data of a cubic Bezier curve in a Geometry Acceleration Structure (GAS) at a given motion time.

data[i] = {x,y,z,w} with {x,y,z} the position and w the radius of control vertex i.

Available in AH, CH

5.1.2.20 optixGetCubicBezierVertexData() [2/2]

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

Deprecated. Call either [optixGetCubicBezierVertexData\(float4 data\[4\]\)](#) for current hit data, or [optixGetCubicBezierVertexDataFromHandle\(\)](#) for random access sphere data.

Returns 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

5.1.2.21 optixGetCubicBezierVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetCubicBezierVertexDataFromHandle (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
```

```
float4 data[4] ) [static]
```

Returns 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

5.1.2.22 optixGetCubicBSplineRocapsVertexData()

```
static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexData (
    float4 data[4] ) [static]
```

5.1.2.23 optixGetCubicBSplineRocapsVertexDataFromHandle()

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

5.1.2.24 optixGetCubicBSplineVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetCubicBSplineVertexData (
    float4 data[4] ) [static]
```

Returns the object space curve control vertex data of a cubic BSpline curve in a Geometry Acceleration Structure (GAS) at a given motion time.

`data[i] = {x,y,z,w}` with `{x,y,z}` the position and `w` the radius of control vertex `i`.

Available in AH, CH

5.1.2.25 optixGetCubicBSplineVertexData() [2/2]

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

Deprecated. Call either `optixGetCubicBSplineVertexData(float4 data[4])` for current hit sphere data, or `optixGetCubicBSplineVertexDataFromHandle()` for random access sphere data.

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`.

$\text{data}[i] = \{x, y, z, w\}$ 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

5.1.2.26 `optixGetCubicBSplineVertexDataFromHandle()`

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

Returns 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`.

$\text{data}[i] = \{x, y, z, w\}$ 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

5.1.2.27 `optixGetCurveParameter()`

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

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

Available in AH, CH

5.1.2.28 `optixGetExceptionCode()`

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

Returns the exception code.

Available in EX

5.1.2.29 `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

5.1.2.30 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

5.1.2.31 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

5.1.2.32 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

5.1.2.33 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

5.1.2.34 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

5.1.2.35 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

5.1.2.36 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

5.1.2.37 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

5.1.2.38 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

5.1.2.39 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

5.1.2.40 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

5.1.2.41 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

5.1.2.42 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

5.1.2.43 `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

5.1.2.44 `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

5.1.2.45 `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

5.1.2.46 `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

5.1.2.47 `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

5.1.2.48 `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

5.1.2.49 `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

5.1.2.50 `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

5.1.2.51 `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

5.1.2.52 `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

5.1.2.53 `optixGetLinearCurveVertexData()` [1/2]

```
static __forceinline__ __device__ void optixGetLinearCurveVertexData (
    float4 data[2] ) [static]
```

Returns the object space control vertex data of the currently intersected linear curve at the current ray time.

Similar to the random access variant `optixGetLinearCurveVertexDataFromHandle`, but does not require setting flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS` when building the corresponding GAS.

It is only valid to call this function if the return value of `optixGetPrimitiveType(optixGetHitKind())` equals `OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR`.

Available in AH, CH

5.1.2.54 `optixGetLinearCurveVertexData()` [2/2]

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

Deprecated. Call either `optixGetLinearCurveVertexData(float4 data[2])` for a current-hit data fetch, or `optixGetLinearCurveVertexDataFromHandle(...)` for a random-access data fetch.

Returns 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

5.1.2.55 `optixGetLinearCurveVertexDataFromHandle()`

```
static __forceinline__ __device__ void
optixGetLinearCurveVertexDataFromHandle (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
```

```
float4 data[2] ) [static]
```

Performs a random access fetch of 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 of any curve, the GAS must be built using the flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS`. If only the vertex data of a currently intersected linear curve is required, it is recommended to use function `optixGetLinearCurveVertexData`. A data fetch of the currently hit primitive does NOT require building the corresponding GAS with 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

5.1.2.56 `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

5.1.2.57 `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

5.1.2.58 `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

5.1.2.59 `optixGetObjectToWorldTransformMatrix()` [1/2]

```
template<typename HitState >
static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix
(
    const HitState & hs,
    float m[12] ) [static]
```

Returns the object-to-world transformation matrix resulting from the transformation list of the templated hit object (see `optixGetWorldToObjectTransformMatrix` for example usage).

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

Available in IS, AH, CH

5.1.2.60 optixGetObjectToWorldTransformMatrix() [2/2]

```
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

5.1.2.61 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

5.1.2.62 optixGetPayload_1()

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

5.1.2.63 optixGetPayload_10()

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

5.1.2.64 optixGetPayload_11()

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

5.1.2.65 optixGetPayload_12()

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

5.1.2.66 optixGetPayload_13()

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

5.1.2.67 optixGetPayload_14()

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

5.1.2.68 optixGetPayload_15()

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

5.1.2.69 optixGetPayload_16()

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

5.1.2.70 optixGetPayload_17()

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

5.1.2.71 optixGetPayload_18()

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

5.1.2.72 optixGetPayload_19()

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

5.1.2.73 optixGetPayload_2()

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

5.1.2.74 optixGetPayload_20()

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

5.1.2.75 optixGetPayload_21()

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

5.1.2.76 optixGetPayload_22()

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

5.1.2.77 optixGetPayload_23()

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

5.1.2.78 optixGetPayload_24()

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

5.1.2.79 optixGetPayload_25()

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

5.1.2.80 optixGetPayload_26()

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

5.1.2.81 optixGetPayload_27()

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

5.1.2.82 optixGetPayload_28()

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

5.1.2.83 optixGetPayload_29()

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

5.1.2.84 `optixGetPayload_3()`

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

5.1.2.85 `optixGetPayload_30()`

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

5.1.2.86 `optixGetPayload_31()`

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

5.1.2.87 `optixGetPayload_4()`

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

5.1.2.88 `optixGetPayload_5()`

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

5.1.2.89 `optixGetPayload_6()`

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

5.1.2.90 `optixGetPayload_7()`

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

5.1.2.91 `optixGetPayload_8()`

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

5.1.2.92 `optixGetPayload_9()`

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

5.1.2.93 `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

5.1.2.94 `optixGetPrimitiveType() [1/2]`

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

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

Available in AH, CH

5.1.2.95 `optixGetPrimitiveType()` [2/2]

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

Function interpreting the result of `optixGetHitKind()`.

Available in all OptiX program types

5.1.2.96 `optixGetQuadraticBSplineRocapsVertexData()`

```
static __forceinline__ __device__ void
optixGetQuadraticBSplineRocapsVertexData (
    float4 data[3] ) [static]
```

5.1.2.97 `optixGetQuadraticBSplineRocapsVertexDataFromHandle()`

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

5.1.2.98 `optixGetQuadraticBSplineVertexData()` [1/2]

```
static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (
    float4 data[3] ) [static]
```

Returns the object space curve control vertex data of a quadratic BSpline curve in a Geometry Acceleration Structure (GAS) at a given motion time.

`data[i] = {x,y,z,w}` with `{x,y,z}` the position and `w` the radius of control vertex `i`.

Available in AH, CH

5.1.2.99 `optixGetQuadraticBSplineVertexData()` [2/2]

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

Returns 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

5.1.2.100 `optixGetQuadraticBSplineVertexDataFromHandle()`

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

Returns 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

5.1.2.101 `optixGetRayFlags()`

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

Returns the rayFlags passed into `optixTrace`.

Available in IS, AH, CH, MS

5.1.2.102 `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

5.1.2.103 `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

5.1.2.104 optixGetRayTmin()

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

Returns the tmin passed into optixTrace.

Available in IS, AH, CH, MS

5.1.2.105 optixGetRayVisibilityMask()

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

Returns the visibilityMask passed into optixTrace.

Available in IS, AH, CH, MS

5.1.2.106 optixGetRemainingTraceDepth()

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

If non-zero it is legal to call optixTrace or optixTraverse without triggering an OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED exception. In the case of optixTrace it represents the number of recursive calls that are remaining and counts down.

Value is in the range of [0..OptixPipelineLinkOptions::maxTraceDepth], and maxTraceDepth has a maximum value of 31.

Available in RG, CH, MS, CC, DC

5.1.2.107 optixGetRibbonNormal() [1/2]

```
static __forceinline__ __device__ float3 optixGetRibbonNormal (
    float2 ribbonParameters ) [static]
```

Return ribbon normal at intersection reported by optixReportIntersection.

Available in AH, CH

5.1.2.108 optixGetRibbonNormal() [2/2]

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

Deprecated. Call either [optixGetRibbonNormal\(float2 ribbonParameters\)](#) for current hit data, or [optixGetRibbonNormalFromHandle\(\)](#) for random access.

Returns ribbon normal at intersection reported by optixReportIntersection.

Available in all OptiX program types

5.1.2.109 optixGetRibbonNormalFromHandle()

```
static __forceinline__ __device__ float3 optixGetRibbonNormalFromHandle (
```

```

    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
    float2 ribbonParameters ) [static]

```

Returns ribbon normal at intersection reported by `optixReportIntersection`.

Available in all OptiX program types

5.1.2.110 `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

5.1.2.111 `optixGetRibbonVertexData()` [1/2]

```

static __forceinline__ __device__ void optixGetRibbonVertexData (
    float4 data[3] ) [static]

```

Returns the object space curve control vertex data of a ribbon (flat quadratic BSpline) in a Geometry Acceleration Structure (GAS) at a given motion time.

`data[i] = {x,y,z,w}` with `{x,y,z}` the position and `w` the radius of control vertex `i`.

Available in AH, CH

5.1.2.112 `optixGetRibbonVertexData()` [2/2]

```

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

```

Deprecated. Call either `optixGetRibbonVertexData(float4 data[3])` for current hit data, or `optixGetRibbonVertexDataFromHandle()` for random access.

Returns 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

5.1.2.113 optixGetRibbonVertexDataFromHandle()

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

Returns 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

5.1.2.114 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

5.1.2.115 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

5.1.2.116 optixGetSphereData() [1/2]

```
static __forceinline__ __device__ void optixGetSphereData (
    float4 data[1] ) [static]
```

Returns the object space sphere data of the currently intersected sphere at the current ray time.

Similar to the random access variant `optixGetSphereDataFromHandle`, but does not require setting flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS` when building the corresponding GAS.

It is only valid to call this function if the return value of `optixGetPrimitiveType(optixGetHitKind())` equals `OPTIX_PRIMITIVE_TYPE_SPHERE`.

Available in AH, CH

5.1.2.117 `optixGetSphereData()` [2/2]

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

Deprecated. Call either `optixGetSphereData(float4 data[1])` for current hit sphere data, or `optixGetSphereDataFromHandle()` for random access sphere data.

Returns 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

5.1.2.118 `optixGetSphereDataFromHandle()`

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

Performs a random access fetch of the object space sphere data, center point and radius, in a Geometry Acceleration Structure (GAS) at a given motion time.

To access vertex data of any curve, the GAS must be built using the flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS`. If only the vertex data of a currently intersected sphere is required, it is recommended to use function `optixGetSphereData`. A data fetch of the currently hit primitive does NOT require building the corresponding GAS with 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

5.1.2.119 `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

5.1.2.120 `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

5.1.2.121 `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

5.1.2.122 `optixGetTransformListSize()`

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

Returns the number of transforms on the current transform list.

Available in IS, AH, CH

5.1.2.123 `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

5.1.2.124 `optixGetTriangleBarycentrics()`

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

Convenience function that returns the first two attributes as floats.

When using `OptixBuildInputTriangleArray` objects, during intersection with a triangle, the barycentric coordinates of the hit are stored into the first two attribute registers.

Available in AH, CH

5.1.2.125 `optixGetTriangleVertexData()` [1/2]

```
static __forceinline__ __device__ void optixGetTriangleVertexData (
```

```
float3 data[3] ) [static]
```

Returns the object space triangle vertex positions of the currently intersected triangle at the current ray time.

Similar to the random access variant `optixGetTriangleVertexDataFromHandle`, but does not require setting flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS` when building the corresponding GAS.

It is only valid to call this function if the return value of `optixGetPrimitiveType(optixGetHitKind())` equals `OPTIX_PRIMITIVE_TYPE_TRIANGLE`.

Available in AH, CH

5.1.2.126 `optixGetTriangleVertexData()` [2/2]

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

[DEPRECATED] Returns the object space triangle vertex positions of a given triangle in a Geometry Acceleration Structure (GAS) at a given motion time. This function is deprecated, use `optixGetTriangleVertexDataFromHandle` for random access triangle vertex data fetch or the overload `optixGetTriangleVertexData(float3 data[3])` for a current triangle hit vertex data fetch.

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

5.1.2.127 `optixGetTriangleVertexDataFromHandle()`

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

Performs a random access data fetch object space vertex position of a given triangle in a Geometry Acceleration Structure (GAS) at a given motion time.

To access vertex data of any triangle, the GAS must be built using the flag `OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS`. If only the vertex data of a currently intersected triangle is required, it is recommended to use function `optixGetTriangleVertexData`. A data fetch of the currently hit primitive does NOT require building the corresponding GAS with 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

5.1.2.128 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

5.1.2.129 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

5.1.2.130 optixGetWorldToObjectTransformMatrix() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix
(
    const HitState & hs,
    float m[12] ) [static]
```

Returns the world-to-object transformation matrix resulting from the transformation list of the templated hit object. Users may implement getRayTime, getTransformListSize, and getTransformListHandle in their own structs, or inherit them from Optix[Incoming|Outgoing]HitObject. Here is an example:

```
struct FixedTimeHitState : OptixIncomingHitObject { float time; forceinline device float getRayTime() {
return time; } }; ... optixGetWorldToObjectTransformMatrix(FixedTimeHitState{ 0.4f }, m);
```

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

Available in IS, AH, CH

5.1.2.131 optixGetWorldToObjectTransformMatrix() [2/2]

```
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

5.1.2.132 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

5.1.2.133 `optixHitObjectGetAttribute_1()`

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

5.1.2.134 `optixHitObjectGetAttribute_2()`

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

5.1.2.135 `optixHitObjectGetAttribute_3()`

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

5.1.2.136 `optixHitObjectGetAttribute_4()`

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

5.1.2.137 `optixHitObjectGetAttribute_5()`

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

5.1.2.138 `optixHitObjectGetAttribute_6()`

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

5.1.2.139 `optixHitObjectGetAttribute_7()`

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

5.1.2.140 `optixHitObjectGetCatmullRomRocapsVertexData()`

```
static __forceinline__ __device__ void
optixHitObjectGetCatmullRomRocapsVertexData (
    float4 data[4] ) [static]
```

5.1.2.141 `optixHitObjectGetCatmullRomVertexData()`

```
static __forceinline__ __device__ void optixHitObjectGetCatmullRomVertexData
(
    float4 data[4] ) [static]
```

Returns the object space curve control vertex data of a CatmullRom spline curve for a valid outgoing

hit object.

$\text{data}[i] = \{x,y,z,w\}$ with $\{x,y,z\}$ the position and w the radius of control vertex i .

It is only valid to call this function if the return value of `optixGetPrimitiveType(optixHitObjectGetHitKind())` equals `OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM`.

Available in RG, CH, MS, CC, DC

5.1.2.142 `optixHitObjectGetClusterId()`

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

Returns the user-provided cluster ID associated with the current outgoing hit object.

Returns `OPTIX_CLUSTER_ID_INVALID` if the closest intersection is not a cluster, or if the hit object is a miss.

see also [OptixPipelineCompileOptions::allowClusteredGeometry](#)

Available in RG, CH, MS, CC, DC

5.1.2.143 `optixHitObjectGetCubicBezierRocapsVertexData()`

```
static __forceinline__ __device__ void
optixHitObjectGetCubicBezierRocapsVertexData (
    float4 data[4] ) [static]
```

5.1.2.144 `optixHitObjectGetCubicBezierVertexData()`

```
static __forceinline__ __device__ void
optixHitObjectGetCubicBezierVertexData (
    float4 data[4] ) [static]
```

Returns the object space curve control vertex data of a cubic Bezier curve for a valid outgoing hit object.

$\text{data}[i] = \{x,y,z,w\}$ with $\{x,y,z\}$ the position and w the radius of control vertex i .

It is only valid to call this function if the return value of `optixGetPrimitiveType(optixHitObjectGetHitKind())` equals `OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER`.

Available in RG, CH, MS, CC, DC

5.1.2.145 `optixHitObjectGetCubicBSplineRocapsVertexData()`

```
static __forceinline__ __device__ void
optixHitObjectGetCubicBSplineRocapsVertexData (
    float4 data[4] ) [static]
```

See [optixHitObjectGetCubicBSplineVertexData](#) for further documentation.

It is only valid to call this function if the return value of `optixGetPrimitiveType(optixHitObjectGetHitKind())` equals `OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE_ROCAPS`.

Available in RG, CH, MS, CC, DC

5.1.2.146 `optixHitObjectGetCubicBSplineVertexData()`

```
static __forceinline__ __device__ void
optixHitObjectGetCubicBSplineVertexData (
    float4 data[4] ) [static]
```

Returns the object space curve control vertex data of a cubic BSpline curve for a valid outgoing hit object.

`data[i] = {x,y,z,w}` with `{x,y,z}` the position and `w` the radius of control vertex `i`.

It is only valid to call this function if the return value of `optixGetPrimitiveType(optixHitObjectGetHitKind())` equals `OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE`.

Available in RG, CH, MS, CC, DC

5.1.2.147 `optixHitObjectGetCurveParameter()`

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

Returns the curve parameter associated with the intersection of a curve.

This function is the hit object's equivalent to `optixGetCurveParameter()`. It is only valid to call this function if the return value of `optixGetPrimitiveType(optixHitObjectGetHitKind())` equals a primitive type that can be used to build an AS with `OptixBuildInputCurveArray` objects.

Available in RG, CH, MS, CC, DC

5.1.2.148 `optixHitObjectGetGASTraversableHandle()`

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

Returns the traversable handle for the Geometry Acceleration Structure (GAS) associated with the current outgoing hit object. Returns 0 if the hit object is not a hit.

Available in RG, CH, MS, CC, DC

5.1.2.149 `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

5.1.2.150 `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

5.1.2.151 `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

5.1.2.152 `optixHitObjectGetLinearCurveVertexData()`

```
static __forceinline__ __device__ void
optixHitObjectGetLinearCurveVertexData (
    float4 data[2] ) [static]
```

Returns the object space control vertex data of the currently intersected linear curve for a valid outgoing hit object. It is the hit object's pendant of [optixGetLinearCurveVertexData\(float4 data\[2\]\)](#).

It is only valid to call this function if the return value of `optixGetPrimitiveType(optixHitObjectGetHitKind())` equals `OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR`.

Available in RG, CH, MS, CC, DC

5.1.2.153 `optixHitObjectGetObjectToWorldTransformMatrix()`

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

Returns the object-to-world transformation matrix resulting from the transformation list of the current outgoing hit object.

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

Available in RG, CH, MS, CC, DC

5.1.2.154 `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

5.1.2.155 `optixHitObjectGetQuadraticBSplineRocapsVertexData()`

```
static __forceinline__ __device__ void
optixHitObjectGetQuadraticBSplineRocapsVertexData (
    float4 data[3] ) [static]
```

5.1.2.156 `optixHitObjectGetQuadraticBSplineVertexData()`

```
static __forceinline__ __device__ void
optixHitObjectGetQuadraticBSplineVertexData (
    float4 data[3] ) [static]
```

Returns the object space curve control vertex data of a quadratic BSpline curve for a valid outgoing hit object.

$data[i] = \{x,y,z,w\}$ with $\{x,y,z\}$ the position and w the radius of control vertex i .

It is only valid to call this function if the return value of `optixGetPrimitiveType(optixHitObjectGetHitKind())` equals `OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE`.

Available in RG, CH, MS, CC, DC

5.1.2.157 `optixHitObjectGetRayFlags()`

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

Returns the rayFlags passed into `optixTrace` associated with the current outgoing hit object.

Available in RG, CH, MS, CC, DC

5.1.2.158 `optixHitObjectGetRayTime()`

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

Returns the rayTime passed into `optixTraverse`, `optixMakeHitObject` or `optixMakeMissHitObject`.

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

Available in RG, CH, MS, CC, DC

5.1.2.159 `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`, `optixMakeHitObject` or `optixMakeMissHitObject`.

Returns 0 for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.160 `optixHitObjectGetRayTmin()`

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

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

Returns 0.0f for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.161 optixHitObjectGetRibbonNormal()

```
static __forceinline__ __device__ float3 optixHitObjectGetRibbonNormal (
    float2 ribbonParameters ) [static]
```

Return ribbon normal at intersection reported by optixReportIntersection.

Available in RG, CH, MS, CC, DC

5.1.2.162 optixHitObjectGetRibbonParameters()

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

Returns the ribbon parameters along directrix (length) and generator (width) of the current curve intersection with primitive type OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE.

This function is the hit object's equivalent to [optixGetRibbonParameters\(\)](#). It is only valid to call this function if the return value of [optixGetPrimitiveType\(optixHitObjectGetHitKind\(\)\)](#) equals OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE.

Available in RG, CH, MS, CC, DC

5.1.2.163 optixHitObjectGetRibbonVertexData()

```
static __forceinline__ __device__ void optixHitObjectGetRibbonVertexData (
    float4 data[3] ) [static]
```

Returns the object space curve control vertex data of a ribbon (flat quadratic BSpline) for a valid outgoing hit object.

data[i] = {x,y,z,w} with {x,y,z} the position and w the radius of control vertex i.

It is only valid to call this function if the return value of [optixGetPrimitiveType\(optixHitObjectGetHitKind\(\)\)](#) equals OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE.

Available in RG, CH, MS, CC, DC

5.1.2.164 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

5.1.2.165 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

5.1.2.166 [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

5.1.2.167 [optixHitObjectGetSphereData\(\)](#)

```
static __forceinline__ __device__ void optixHitObjectGetSphereData (
    float4 data[1] ) [static]
```

Returns the object space sphere data of the currently intersected sphere for a valid outgoing hit object. It is the hit object's pendant of [optixGetSphereData\(float4 data\[1\]\)](#).

It is only valid to call this function if the return value of [optixGetPrimitiveType\(optixHitObjectGetHitKind\(\)\)](#) equals `OPTIX_PRIMITIVE_TYPE_SPHERE`.

Available in RG, CH, MS, CC, DC

5.1.2.168 [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

5.1.2.169 [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

5.1.2.170 [optixHitObjectGetTraverseData\(\)](#)

```
static __forceinline__ __device__ void optixHitObjectGetTraverseData (
    OptixTraverseData * data ) [static]
```

Serializes the current outgoing hit object which allows to recreate it at a later point using [optixMakeHitObject](#).

Parameters

out	data
-----	------

Available in RG, CH, MS, CC, DC

5.1.2.171 optixHitObjectGetTriangleBarycentrics()

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

Returns the barycentric coordinates of the hit point on an intersected triangle.

This function is the hit object's equivalent to [optixGetTriangleBarycentrics\(\)](#). It is only valid to call this function if the return value of [optixGetPrimitiveType\(optixHitObjectGetHitKind\(\)\)](#) equals `OPTIX_PRIMITIVE_TYPE_TRIANGLE`.

Available in RG, CH, MS, CC, DC

5.1.2.172 optixHitObjectGetTriangleVertexData()

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

Returns the object space triangle vertex positions of the intersected triangle for a valid outgoing hit object. It is the hit object's pendant of [optixGetTriangleVertexData\(float3 data\[3\]\)](#).

It is only valid to call this function if the return value of [optixGetPrimitiveType\(optixHitObjectGetHitKind\(\)\)](#) equals `OPTIX_PRIMITIVE_TYPE_TRIANGLE`.

Available in RG, CH, MS, CC, DC

5.1.2.173 optixHitObjectGetWorldRayDirection()

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.174 optixHitObjectGetWorldRayOrigin()

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.175 `optixHitObjectGetWorldToObjectTransformMatrix()`

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

Returns the world-to-object transformation matrix resulting from the transformation list of the current outgoing hit object.

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

Available in RG, CH, MS, CC, DC

5.1.2.176 `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

5.1.2.177 `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

5.1.2.178 `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

5.1.2.179 `optixHitObjectSetSbtRecordIndex()`

```
static __forceinline__ __device__ void optixHitObjectSetSbtRecordIndex (
    unsigned int sbtRecordIndex ) [static]
```

Sets the SBT record index in the current outgoing hit object.

Available in RG, CH, MS, CC, DC

5.1.2.180 `optixHitObjectTransformNormalFromObjectToWorldSpace()`

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

Transforms the normal using object-to-world transformation matrix resulting from the transformation list of the current outgoing hit object.

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

Available in RG, CH, MS, CC, DC

5.1.2.181 optixHitObjectTransformNormalFromWorldToObjectSpace()

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

Transforms the normal using world-to-object transformation matrix resulting from the transformation list of the current outgoing hit object.

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

Available in RG, CH, MS, CC, DC

5.1.2.182 optixHitObjectTransformPointFromObjectToWorldSpace()

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

Transforms the point using object-to-world transformation matrix resulting from the transformation list of the current outgoing hit object.

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

Available in RG, CH, MS, CC, DC

5.1.2.183 optixHitObjectTransformPointFromWorldToObjectSpace()

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

Transforms the point using world-to-object transformation matrix resulting from the transformation list of the current outgoing hit object.

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

Available in RG, CH, MS, CC, DC

5.1.2.184 optixHitObjectTransformVectorFromObjectToWorldSpace()

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

Transforms the vector using object-to-world transformation matrix resulting from the transformation list of the current outgoing hit object.

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

Available in RG, CH, MS, CC, DC

5.1.2.185 optixHitObjectTransformVectorFromWorldToObjectSpace()

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

Transforms the vector using world-to-object transformation matrix resulting from the transformation

list of the current outgoing hit object.

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

Available in RG, CH, MS, CC, DC

5.1.2.186 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

5.1.2.187 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, optixMakeMissHitObject, optixMakeHitObject 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

5.1.2.188 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, optixMakeMissHitObject, optixMakeHitObject or optixMakeNopHitObject.

Parameters

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

Available in RG, CH, MS, CC

5.1.2.189 optixIsBackFaceHit() [1/2]

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

Function interpreting the hit kind associated with the current optixReportIntersection.

Available in AH, CH

5.1.2.190 optixIsBackFaceHit() [2/2]

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

Function interpreting the result of [optixGetHitKind\(\)](#).

Available in all OptiX program types

5.1.2.191 optixIsFrontFaceHit() [1/2]

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

Function interpreting the hit kind associated with the current [optixReportIntersection](#).

Available in AH, CH

5.1.2.192 optixIsFrontFaceHit() [2/2]

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

Function interpreting the result of [optixGetHitKind\(\)](#).

Available in all OptiX program types

5.1.2.193 optixIsTriangleBackFaceHit()

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

Convenience function interpreting the result of [optixGetHitKind\(\)](#).

Available in AH, CH

5.1.2.194 optixIsTriangleFrontFaceHit()

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

Convenience function interpreting the result of [optixGetHitKind\(\)](#).

Available in AH, CH

5.1.2.195 optixIsTriangleHit()

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

Convenience function interpreting the result of [optixGetHitKind\(\)](#).

Available in AH, CH

5.1.2.196 optixMakeHitObject()

```
static __forceinline__ __device__ void optixMakeHitObject (
    OptixTraversableHandle handle,
    float3 rayOrigin,
    float3 rayDirection,
    float tmin,
    float rayTime,
```

```

    unsigned int rayFlags,
    OptixTraverseData traverseData,
    const OptixTraversableHandle * transforms,
    unsigned int numTransforms ) [static]

```

Constructs an outgoing hit object from the hit object data provided. The traverseData needs to be collected from a previous hit object using `optixHitObjectGetTraverseData`. 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>rayTime</i>	
in	<i>rayFlags</i>	really only 16 bits, combination of OptixRayFlags
in	<i>traverseData</i>	
in	<i>transforms</i>	
in	<i>numTransforms</i>	

Available in RG, CH, MS, CC

5.1.2.197 optixMakeMissHitObject()

```

static __forceinline__ __device__ void optixMakeMissHitObject (
    unsigned int missSBTIndex,
    float3 rayOrigin,
    float3 rayDirection,
    float tmin,
    float tmax,
    float rayTime,
    unsigned int rayFlags ) [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>	
in	<i>rayFlags</i>	really only 16 bits, combination of OptixRayFlags

Available in RG, CH, MS, CC

5.1.2.198 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 [optixHitObjectIsNop](#) will return a non-zero result.

Available in RG, CH, MS, CC

5.1.2.199 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

5.1.2.200 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

5.1.2.201 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

5.1.2.202 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

5.1.2.203 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

5.1.2.204 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

5.1.2.205 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

5.1.2.206 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

5.1.2.207 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

5.1.2.208 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

5.1.2.209 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

5.1.2.210 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

5.1.2.211 optixSetPayload_1()

```

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

```

5.1.2.212 optixSetPayload_10()

```

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

```

5.1.2.213 optixSetPayload_11()

```

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

```

5.1.2.214 optixSetPayload_12()

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

5.1.2.215 optixSetPayload_13()

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

5.1.2.216 optixSetPayload_14()

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

5.1.2.217 optixSetPayload_15()

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

5.1.2.218 optixSetPayload_16()

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

5.1.2.219 optixSetPayload_17()

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

5.1.2.220 optixSetPayload_18()

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

5.1.2.221 optixSetPayload_19()

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

5.1.2.222 optixSetPayload_2()

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

5.1.2.223 optixSetPayload_20()

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

5.1.2.224 optixSetPayload_21()

```
static __forceinline__ __device__ void optixSetPayload_21 (  

```

unsigned int *p*) *[static]*

5.1.2.225 optixSetPayload_22()

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

5.1.2.226 optixSetPayload_23()

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

5.1.2.227 optixSetPayload_24()

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

5.1.2.228 optixSetPayload_25()

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

5.1.2.229 optixSetPayload_26()

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

5.1.2.230 optixSetPayload_27()

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

5.1.2.231 optixSetPayload_28()

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

5.1.2.232 optixSetPayload_29()

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

5.1.2.233 optixSetPayload_3()

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

5.1.2.234 optixSetPayload_30()

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

5.1.2.235 `optixSetPayload_31()`

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

5.1.2.236 `optixSetPayload_4()`

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

5.1.2.237 `optixSetPayload_5()`

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

5.1.2.238 `optixSetPayload_6()`

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

5.1.2.239 `optixSetPayload_7()`

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

5.1.2.240 `optixSetPayload_8()`

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

5.1.2.241 `optixSetPayload_9()`

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

5.1.2.242 `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

5.1.2.243 `optixTerminateRay()`

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

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

Available in AH

5.1.2.244 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

5.1.2.245 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

5.1.2.246 optixTexFootprint2DLod()

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

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

Parameters

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

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

5.1.2.247 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

5.1.2.248 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

5.1.2.249 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

5.1.2.250 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

5.1.2.251 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

5.1.2.252 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

5.1.2.253 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

5.1.2.254 `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

5.1.2.255 `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

5.1.2.256 `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

5.1.2.257 optixTrace() [2/2]

```

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

```

Initiates a ray tracing query starting with the given traversable.

Parameters

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

Available in RG, CH, MS, CC

5.1.2.258 optixTransformNormalFromObjectToWorldSpace() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformNormalFromObjectToWorldSpace (
    const HitState & hs,
    float3 normal ) [static]
```

Transforms the normal using object-to-world transformation matrix resulting from the transformation list of the templated hit object (see `optixGetWorldToObjectTransformMatrix` for example usage).

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

Available in IS, AH, CH

5.1.2.259 optixTransformNormalFromObjectToWorldSpace() [2/2]

```
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

5.1.2.260 optixTransformNormalFromWorldToObjectSpace() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformNormalFromWorldToObjectSpace (
    const HitState & hs,
```

```
float3 normal ) [static]
```

Transforms the normal using world-to-object transformation matrix resulting from the transformation list of the templated hit object (see `optixGetWorldToObjectTransformMatrix` for example usage).

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

Available in IS, AH, CH

5.1.2.261 `optixTransformNormalFromWorldToObjectSpace()` [2/2]

```
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

5.1.2.262 `optixTransformPointFromObjectToWorldSpace()` [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformPointFromObjectToWorldSpace (
    const HitState & hs,
    float3 point ) [static]
```

Transforms the point using object-to-world transformation matrix resulting from the transformation list of the templated hit object (see `optixGetWorldToObjectTransformMatrix` for example usage).

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

Available in IS, AH, CH

5.1.2.263 `optixTransformPointFromObjectToWorldSpace()` [2/2]

```
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

5.1.2.264 `optixTransformPointFromWorldToObjectSpace()` [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformPointFromWorldToObjectSpace (
    const HitState & hs,
    float3 point ) [static]
```

Transforms the point using world-to-object transformation matrix resulting from the transformation list of the templated hit object (see `optixGetWorldToObjectTransformMatrix` for example usage).

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

Available in IS, AH, CH

5.1.2.265 `optixTransformPointFromWorldToObjectSpace()` [2/2]

```
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

5.1.2.266 `optixTransformVectorFromObjectToWorldSpace()` [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformVectorFromObjectToWorldSpace (
    const HitState & hs,
    float3 vec ) [static]
```

Transforms the vector using object-to-world transformation matrix resulting from the transformation list of the templated hit object (see `optixGetWorldToObjectTransformMatrix` for example usage).

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

Available in IS, AH, CH

5.1.2.267 `optixTransformVectorFromObjectToWorldSpace()` [2/2]

```
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

5.1.2.268 `optixTransformVectorFromWorldToObjectSpace()` [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformVectorFromWorldToObjectSpace (
    const HitState & hs,
    float3 vec ) [static]
```

Transforms the vector using world-to-object transformation matrix resulting from the transformation list of the templated hit object (see `optixGetWorldToObjectTransformMatrix` for example usage).

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

Available in IS, AH, CH

5.1.2.269 optixTransformVectorFromWorldToObjectSpace() [2/2]

```
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

5.1.2.270 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

Parameters

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

5.1.2.271 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>	
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

Parameters

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

Available in RG, CH, MS, CC, DC

5.1.2.272 `optixUndefinedValue()`

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

Returns an undefined value.

Available anywhere

5.2 Cooperative Vector

Classes

- class `OptixCoopVec< T, N >`

Functions

- `template<typename VecTOut >`
`static __forceinline__ __device__ VecTOut optixCoopVecLoad (CUdeviceptr ptr)`
- `template<typename VecTOut , typename T >`
`static __forceinline__ __device__ VecTOut optixCoopVecLoad (T *ptr)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecExp2 (const VecT &vec)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecLog2 (const VecT &vec)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecTanh (const VecT &vec)`
- `template<typename VecTOut , typename VecTIn >`
`static __forceinline__ __device__ VecTOut optixCoopVecCvt (const VecTIn &vec)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecMin (const VecT &vecA, const VecT &vecB)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecMin (const VecT &vecA, typename VecT ::value_type B)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecMax (const VecT &vecA, const VecT &vecB)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecMax (const VecT &vecA, typename VecT ::value_type B)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecMul (const VecT &vecA, const VecT &vecB)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecAdd (const VecT &vecA, const VecT &vecB)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecSub (const VecT &vecA, const VecT &vecB)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecStep (const VecT &vecA, const VecT &vecB)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecFFMA (const VecT &vecA, const VecT &vecB, const VecT &vecC)`

- `template<typename VecTOut , typename VecTIn , OptixCoopVecElemType inputInterpretation, OptixCoopVecMatrixLayout matrixLayout, bool transpose, unsigned int N, unsigned int K, OptixCoopVecElemType matrixElementType, OptixCoopVecElemType biasElementType>`
`static __forceinline__ __device__ VecTOut optixCoopVecMatMul (const VecTIn &inputVector,`
`CUdeviceptr matrix, unsigned matrixOffsetInBytes, CUdeviceptr bias, unsigned`
`biasOffsetInBytes, unsigned rowColumnStrideInBytes=0)`
- `template<typename VecTOut , typename VecTIn , OptixCoopVecElemType inputInterpretation, OptixCoopVecMatrixLayout matrixLayout, bool transpose, unsigned int N, unsigned int K, OptixCoopVecElemType matrixElementType>`
`static __forceinline__ __device__ VecTOut optixCoopVecMatMul (const VecTIn &inputVector,`
`CUdeviceptr matrix, unsigned matrixOffsetInBytes, unsigned rowColumnStrideInBytes=0)`
- `template<typename VecTIn >`
`static __forceinline__ __device__ void optixCoopVecReduceSumAccumulate (const VecTIn`
`&inputVector, CUdeviceptr outputVector, unsigned offsetInBytes)`
- `template<typename VecTA , typename VecTB , OptixCoopVecMatrixLayout matrixLayout =`
`OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL>`
`static __forceinline__ __device__ void optixCoopVecOuterProductAccumulate (const VecTA`
`&vecA, const VecTB &vecB, CUdeviceptr outputMatrix, unsigned offsetInBytes, unsigned`
`rowColumnStrideInBytes=0)`
- `template<unsigned int N, unsigned int K, OptixCoopVecElemType elementType,`
`OptixCoopVecMatrixLayout layout = OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCEING_`
`OPTIMAL, unsigned int rowColumnStrideInBytes = 0>`
`static __forceinline__ __device__ unsigned int optixCoopVecGetMatrixSize ()`

5.2.1 Detailed Description

5.2.2 Function Documentation

5.2.2.1 optixCoopVecAdd()

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecAdd (
    const VecT & vecA,
    const VecT & vecB ) [static]
```

Available anywhere.

5.2.2.2 optixCoopVecCvt()

```
template<typename VecTOut , typename VecTIn >
static __forceinline__ __device__ VecTOut optixCoopVecCvt (
    const VecTIn & vec ) [static]
```

Convert from VecTIn to VecTOut. Not all conversions are supported, only integral to 16 or 32-bit floating point.

Available anywhere

5.2.2.3 optixCoopVecExp2()

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecExp2 (
    const VecT & vec ) [static]
```

Following functions are designed to facilitate activation function evaluation between calls to `optixCoopVecMatMul`. Utilizing only these functions on the activation vectors will typically improve performance.

Available anywhere

5.2.2.4 `optixCoopVecFFMA()`

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecFFMA (
    const VecT & vecA,
    const VecT & vecB,
    const VecT & vecC ) [static]
```

Available anywhere.

5.2.2.5 `optixCoopVecGetMatrixSize()`

```
template<unsigned int N, unsigned int K, OptixCoopVecElemType elementType,
OptixCoopVecMatrixLayout layout = OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_
OPTIMAL, unsigned int rowColumnStrideInBytes = 0>
static __forceinline__ __device__ unsigned int optixCoopVecGetMatrixSize ( )
[static]
```

This function is intended strictly for matrix layouts that must be computed through the host API, `optixCoopVecMatrixComputeSize`, but is needed on the device. For optimal performance the offsets to each layer in a network should be constant, so this function can be used to facilitate calculating the offset for subsequent layers in shader code. It can also be used for calculating the size of row and column major matrices, but the `rowColumnStrideInBytes` template parameter must be specified, so that it can be calculated during compilation.

For row and column ordered matrix layouts, when `rowColumnStrideInBytes` is 0, the stride will assume tight packing.

Results will be rounded to the next multiple of 64 to make it easy to pack the matrices in memory and have the correct alignment.

Results are in number of bytes, and should match the output of the host function `optixCoopVecMatrixComputeSize`.

Template Parameters

<i>N,K</i>	dimensions of the matrix
<i>elementType</i>	Type of the matrix elements
<i>layout</i>	Layout of the matrix

Available anywhere

5.2.2.6 `optixCoopVecLoad()` [1/2]

```
template<typename VecTOut >
static __forceinline__ __device__ VecTOut optixCoopVecLoad (
    CUdeviceptr ptr ) [static]
```

Load the vector from global memory. The memory address must be 16 byte aligned regardless of the

type and number of elements in the vector.

Available anywhere

5.2.2.7 optixCoopVecLoad() [2/2]

```
template<typename VecTOut , typename T >
static __forceinline__ __device__ VecTOut optixCoopVecLoad (
    T * ptr ) [static]
```

Load the vector from global memory. The memory address must be 16 byte aligned regardless of the type and number of elements in the vector.

Available anywhere

5.2.2.8 optixCoopVecLog2()

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecLog2 (
    const VecT & vec ) [static]
```

Available anywhere.

5.2.2.9 optixCoopVecMatMul() [1/2]

```
template<typename VecTOut , typename VecTIn , OptixCoopVecElemType
inputInterpretation, OptixCoopVecMatrixLayout matrixLayout, bool transpose,
unsigned int N, unsigned int K, OptixCoopVecElemType matrixElementType,
OptixCoopVecElemType biasElementType>
static __forceinline__ __device__ VecTOut optixCoopVecMatMul (
    const VecTIn & inputVector,
    CUdeviceptr matrix,
    unsigned matrixOffsetInBytes,
    CUdeviceptr bias,
    unsigned biasOffsetInBytes,
    unsigned rowColumnStrideInBytes = 0 ) [static]
```

Computes a vector matrix multiplication with an addition of a bias.

$$\begin{matrix} A * B & + C & = D \\ \text{Does matrix * inputVector + bias = output} \\ [N \times K] & [K \times 1] & [N \times 1] = [N \times 1] \end{matrix}$$

Not all combinations of inputType and matrixElementType are supported. See the following table for supported configurations.

FLOAT16	FLOAT16	FLOAT16	FLOAT16	FLOAT16
FLOAT16	FLOAT8_E4M3	FLOAT8_E4M3	FLOAT16	FLOAT16
FLOAT16	FLOAT8_E5M4	FLOAT8_E5M4	FLOAT16	FLOAT16
FLOAT16	UINT8/INT8	UINT8/INT8	UINT32/INT32	UINT32/INT32
FLOAT32	UINT8/INT8	UINT8/INT8	UINT32/INT32	UINT32/INT32
UINT8/INT8	UINT8/INT8	UINT8/INT8	UINT32/INT32	UINT32/INT32

If either the input or matrix is signed, then the bias and output must also be signed.

When `matrixElementType` is `OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E4M3` or `OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E5M2` the `matrixLayout` must be either `OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL` or `OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL`.

When the `inputVector`'s element type does not match the `inputInterpretation` arithmetically casting is performed on the input values to match the `inputInterpretation`.

If `transpose` is true, the matrix is treated as being stored transposed in memory (stored as $K \times N$ instead of $N \times K$). Set other parameters as if the matrix was not transposed in memory. Not all matrix element types or matrix layouts support transpose. Only `OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16` is supported. Only `OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL` and `OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL` are supported.

The bias pointer is assumed to not be null and may be dereferenced. If you wish to do the matrix multiply without a bias then use the overloaded version of this function that does not take the bias.

For row and column ordered matrix layouts, the stride will assume tight packing when `rowColumnStrideInBytes` is a constant immediate 0 (computed values or loaded from memory will not work). Ignored for other matrix layouts. Value must be 16 byte aligned.

Template Parameters

<i>VecTOut</i>	Type must match <code>biasElementType</code> and size must match <code>N</code>
<i>VecTIn</i>	Type must be <code>i32</code> , <code>f16</code> or <code>f32</code> type and size must match <code>K</code>
<i>inputInterpretation</i>	Must match <code>matrixLayout</code>
<i>matrixLayout</i>	The layout of the matrix in memory
<i>transpose</i>	Whether the data in memory for matrix is transposed from the specified layout
<i>N</i>	Must match <code>VecTOut::size</code>
<i>K</i>	Must match <code>VecTIn::size</code>
<i>matrixElementType</i>	Type of elements stored in memory
<i>biasElementType</i>	Type of elements stored in memory, must also match <code>VecTOut::elementType</code>

Parameters

in	<i>inputVector</i>	
in	<i>matrix</i>	pointer to global memory. Array of $N \times K$ elements. 64 byte aligned. Must not be modified during use.
in	<i>matrixOffsetInBytes</i>	offset to start of matrix data. Using the same value for matrix with different offsets for all layers yields more efficient execution. 64 byte aligned.
in	<i>bias</i>	pointer to global memory. Array of <code>N</code> elements. 16 byte aligned. Must not be modified during use.
in	<i>biasOffsetInBytes</i>	offset to start of bias data. Using the same value for bias with different offsets for all layers yields more efficient execution. 16 byte aligned.
in	<i>rowColumnStrideInBytes</i>	for row or column major matrix layouts, this identifies the stride between columns or rows.

Available in all OptiX program types

5.2.2.10 optixCoopVecMatMul() [2/2]

```
template<typename VecTOut , typename VecTIn , OptixCoopVecElemType
inputInterpretation, OptixCoopVecMatrixLayout matrixLayout, bool transpose,
unsigned int N, unsigned int K, OptixCoopVecElemType matrixElementType>
static __forceinline__ __device__ VecTOut optixCoopVecMatMul (
    const VecTIn & inputVector,
    CUdeviceptr matrix,
    unsigned matrixOffsetInBytes,
    unsigned rowColumnStrideInBytes = 0 ) [static]
```

Same as `optixCoopVecMatMul`, but without the bias parameters.

5.2.2.11 optixCoopVecMax() [1/2]

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecMax (
    const VecT & vecA,
    const VecT & vecB ) [static]
```

Available anywhere.

5.2.2.12 optixCoopVecMax() [2/2]

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecMax (
    const VecT & vecA,
    typename VecT::value_type B ) [static]
```

Available anywhere.

5.2.2.13 optixCoopVecMin() [1/2]

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecMin (
    const VecT & vecA,
    const VecT & vecB ) [static]
```

Available anywhere.

5.2.2.14 optixCoopVecMin() [2/2]

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecMin (
    const VecT & vecA,
    typename VecT::value_type B ) [static]
```

Available anywhere.

5.2.2.15 optixCoopVecMul()

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecMul (
    const VecT & vecA,
    const VecT & vecB ) [static]
```

Available anywhere.

5.2.2.16 optixCoopVecOuterProductAccumulate()

```
template<typename VecTA , typename VecTB , OptixCoopVecMatrixLayout
matrixLayout = OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL>
static __forceinline__ __device__ void optixCoopVecOuterProductAccumulate (
    const VecTA & vecA,
    const VecTB & vecB,
    CUdeviceptr outputMatrix,
    unsigned offsetInBytes,
    unsigned rowColumnStrideInBytes = 0 ) [static]
```

Produces a matrix outer product of the input `vecA` and `vecB` (`vecA * transpose(vecB)`) and does a component-wise atomic add reduction of the result into global memory starting `offsetInBytes` bytes after `outputMatrix`. The dimensions of the matrix are `[VecTA::size, VecTB::size]`. `VecTA::elementType`, `VecTB::elementType` and the element type of the matrix must be the same, no type conversion is performed. The element type must be `OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16`.

`outputMatrix + offsetInBytes` must be 4B aligned, but performance may be better with 128 byte alignments.

The output matrix will be in `matrixLayout` layout, though currently only `OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL` layout is supported.

Template Parameters

<i>VecTA</i>	Type of <code>vecA</code>
<i>VecTB</i>	Type of <code>vecB</code>
<i>matrixLayout</i>	Layout of matrix stored in <code>outputMatrix</code>

Parameters

in	<i>vecA</i>	
in	<i>vecB</i>	
in	<i>outputMatrix</i>	pointer to global memory on the device, sum with <i>offsetInBytes</i> must be a multiple of 4
in	<i>offsetInBytes</i>	offset in bytes from <i>outputMatrix</i> , sum with <i>outputMatrix</i> must be a multiple of 4
in	<i>rowColumnStrideInBytes</i>	stride between rows or columns, zero takes natural stride, ignored for optimal layouts

Available in all OptiX program types

5.2.2.17 optixCoopVecReduceSumAccumulate()

```
template<typename VecTIn >
static __forceinline__ __device__ void optixCoopVecReduceSumAccumulate (
    const VecTIn & inputVector,
    CUdeviceptr outputVector,
    unsigned offsetInBytes ) [static]
```

Performs a component-wise atomic add reduction of the vector into global memory starting at *offsetInBytes* bytes after *outputVector*.

VecTIn::elementType must be of type `OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16` or `OPTIX_COOP_VEC_ELEM_TYPE_FLOAT32`. The memory backed by *outputVector* + *offsetInBytes* must be large enough to accomodate *VecTIn::size* elements. The type of data in *outputVector* must match *VecTIn::elementType*. No type conversion is performed. *outputVector* + *offsetInBytes* must be 4 byte aligned.

Template Parameters

<i>VecTIn</i>	Type of <i>inputVector</i>
---------------	----------------------------

Parameters

in	<i>inputVector</i>	
in	<i>outputVector</i>	pointer to global memory on the device, sum with <i>offsetInBytes</i> must be a multiple of 4
in	<i>offsetInBytes</i>	offset in bytes from <i>outputVector</i> , sum with <i>outputVector</i> must be a multiple of 4

Available in all OptiX program types

5.2.2.18 optixCoopVecStep()

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecStep (
    const VecT & vecA,
    const VecT & vecB ) [static]
```

Returns $\text{result}[i] = (\text{vecA}[i] < \text{vecB}[i]) ? 0 : 1;$.

Available anywhere

5.2.2.19 optixCoopVecSub()

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecSub (
    const VecT & vecA,
    const VecT & vecB ) [static]
```

Available anywhere.

5.2.2.20 optixCoopVecTanh()

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecTanh (
```

```
const VecT & vec ) [static]
```

Available anywhere.

5.3 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`

5.3.1 Detailed Description

OptiX Function Table.

5.3.2 Macro Definition Documentation

5.3.2.1 OPTIX_CONCATENATE_ABI_VERSION

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

5.3.2.2 OPTIX_CONCATENATE_ABI_VERSION_IMPL

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

5.3.2.3 OPTIX_FUNCTION_TABLE_SYMBOL

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

5.3.3 Typedef Documentation

5.3.3.1 OptixFunctionTable

```
typedef struct OptixFunctionTable OptixFunctionTable
```

The function table containing all API functions.

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

5.3.4 Variable Documentation

5.3.4.1 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`.

5.4 Host API

Modules

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

5.4.1 Detailed Description

OptiX Host API.

5.5 Error handling

Functions

- `OPTIXAPI` `const char *` `optixGetErrorName` (`OptixResult` `result`)
- `OPTIXAPI` `const char *` `optixGetErrorString` (`OptixResult` `result`)

5.5.1 Detailed Description

5.5.2 Function Documentation

5.5.2.1 `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](#)

5.5.2.2 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](#)

5.6 Device context

Functions

- 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)

5.6.1 Detailed Description

5.6.2 Function Documentation

5.6.2.1 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

5.6.2.2 `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.

5.6.2.3 `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

5.6.2.4 `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

5.6.2.5 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 <i>locationSize</i> > 0
in	<i>locationSize</i>	<i>locationSize</i>

5.6.2.6 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

5.6.2.7 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

5.6.2.8 `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

5.6.2.9 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

5.6.2.10 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

5.7 Pipelines

Functions

- `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 optixPipelineSetStackSizeFromCallDepths` (`OptixPipeline` pipeline, unsigned int maxTraceDepth, unsigned int maxContinuationCallableDepth, unsigned int maxDirectCallableDepthFromState, unsigned int maxDirectCallableDepthFromTraversal, unsigned int maxTraversableGraphDepth)
- `OPTIXAPI OptixResult optixPipelineSetStackSize` (`OptixPipeline` pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)
- `OPTIXAPI OptixResult optixPipelineSymbolMemcpyAsync` (`OptixPipeline` pipeline, const char *name, void *mem, size_t sizeInBytes, size_t offsetInBytes, `OptixPipelineSymbolMemcpyKind` kind, `CUstream` stream)

5.7.1 Detailed Description

5.7.2 Function Documentation

5.7.2.1 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	context	
in	pipelineCompileOptions	
in	pipelineLinkOptions	

Parameters

in	<i>programGroups</i>	array of ProgramGroup objects
in	<i>numProgramGroups</i>	number of ProgramGroup objects
out	<i>logString</i>	Information will be written to this string. If logStringSize > 0 logString will be null terminated.
in, out	<i>logStringSize</i>	
out	<i>pipeline</i>	

5.7.2.2 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.

5.7.2.3 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	<i>pipeline</i>	The pipeline to configure the stack size for.
in	<i>directCallableStackSizeFromTraversal</i>	The direct stack size requirement for direct callables invoked from IS or AH.
in	<i>directCallableStackSizeFromState</i>	The direct stack size requirement for direct callables invoked from RG, MS, or CH.
in	<i>continuationStackSize</i>	The continuation stack requirement.

Parameters

in	<i>maxTraversableGraphDepth</i>	The maximum depth of a traversable graph passed to trace.
----	---------------------------------	---

5.7.2.4 `optixPipelineSetStackSizeFromCallDepths()`

```

OPTIXAPI OptixResult optixPipelineSetStackSizeFromCallDepths (
    OptixPipeline pipeline,
    unsigned int maxTraceDepth,
    unsigned int maxContinuationCallableDepth,
    unsigned int maxDirectCallableDepthFromState,
    unsigned int maxDirectCallableDepthFromTraversal,
    unsigned int maxTraversableGraphDepth )

```

Sets the stack size for a pipeline based on the given depth parameters.

When the pipeline is created the stack sizes for a pipeline are configured based on the depth values that were given in the [OptixPipelineLinkOptions](#). This method allows to reconfigure the pipeline to new values for the maximum trace depth and the maximum callable depths which includes a recalculation of the stack sizes.

Parameters

in	<i>pipeline</i>	The pipeline to set the stack size for.
in	<i>maxTraceDepth</i>	The maximum trace recursion depth. See OptixPipelineLinkOptions::maxTraceDepth .
in	<i>maxContinuationCallableDepth</i>	The maximum depth of continuation callable call graphs. See OptixPipelineLinkOptions::maxContinuationCallableDepth .
in	<i>maxDirectCallableDepthFromState</i>	The maximum depth of direct callable call graphs called from RG, CH, MS or CC. See OptixPipelineLinkOptions::maxDirectCallableDepthFromState .
in	<i>maxDirectCallableDepthFromTraversal</i>	The maximum depth of direct callable call graphs called from IS or AH. See OptixPipelineLinkOptions::maxDirectCallableDepthFromTraversal .
in	<i>maxTraversableGraphDepth</i>	The maximum depth of a traversable graph passed to trace.

5.7.2.5 `optixPipelineSymbolMemcpyAsync()`

```

OPTIXAPI OptixResult optixPipelineSymbolMemcpyAsync (
    OptixPipeline pipeline,
    const char * name,
    void * mem,
    size_t sizeInBytes,
    size_t offsetInBytes,

```

```
OptixPipelineSymbolMemcpyKind kind,
CUstream stream )
```

Copies data from or to a global symbol in the pipeline. Depending on the given kind of the copy operation, the mem parameter acts as the source or the target of the operation. The sizeInBytes parameter determines how many bytes are copied. The offsetInBytes parameter determines the offset in bytes in the target memory.

Parameters

in	<i>pipeline</i>	The pipeline to get the symbol address from/to.
in	<i>name</i>	The name of the symbol to copy data from/to.
in	<i>mem</i>	The memory where to copy data from/to. Depending on the kind of the copy operation this is either a host or a device pointer.
in	<i>sizeInBytes</i>	The amount of bytes to copy.
in	<i>offsetInBytes</i>	The offset in the symbol's memory to copy the data from/to.
in	<i>kind</i>	A flag that determines the direction of the copy operation.
in	<i>stream</i>	The CUstream to execute the asynchronous operation in.

5.8 Modules

Functions

- 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 optixModuleCancelCreation (OptixModule module, OptixCreationFlags flags)`
- OPTIXAPI `OptixResult optixDeviceContextCancelCreations (OptixDeviceContext context, OptixCreationFlags flags)`
- OPTIXAPI `OptixResult optixModuleDestroy (OptixModule module)`
- OPTIXAPI `OptixResult optixBuiltinISModuleGet (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixBuiltinISOOptions *builtinISOOptions, OptixModule *builtinModule)`

5.8.1 Detailed Description

5.8.2 Function Documentation

5.8.2.1 optixBuiltinISModuleGet()

```
OPTIXAPI OptixResult optixBuiltinISModuleGet (
    OptixDeviceContext context,
```

```

const OptixModuleCompileOptions * moduleCompileOptions,
const OptixPipelineCompileOptions * pipelineCompileOptions,
const OptixBuiltinISOOptions * builtinISOOptions,
OptixModule * builtinModule )

```

Returns a module containing the intersection program for the built-in primitive type specified by the builtinISOOptions. This module must be used as the moduleIS for the OptixProgramGroupHitgroup in any SBT record for that primitive type. (The entryFunctionNameIS should be null.)

5.8.2.2 optixDeviceContextCancelCreations()

```

OPTIXAPI OptixResult optixDeviceContextCancelCreations (
    OptixDeviceContext context,
    OptixCreationFlags flags )

```

Used to cancel creation of all modules associated with an OptixDeviceContext. Conditionally blocks (see OptixCreationFlags)

Thread safety: Safe to call from any thread

5.8.2.3 optixModuleCancelCreation()

```

OPTIXAPI OptixResult optixModuleCancelCreation (
    OptixModule module,
    OptixCreationFlags flags )

```

Used to cancel task-based module creation. A canceled module will transition to OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE if there are unfinished tasks that have been returned to the user, or OPTIX_MODULE_COMPILE_STATE_FAILED if all returned tasks have finished executing, at which point it should be treated as any other module that has failed compilation. The user may continue executing tasks of a canceled module, they will simply return OPTIX_ERROR_CREATION_CANCELED without performing any compilation and without creating new tasks.

Conditionally blocks (see OptixCreationFlags)

Thread safety: Safe to call from any thread

5.8.2.4 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 <code>inputSize</code> 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 <code>logStringSize > 0</code> <code>logString</code> 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.

5.8.2.5 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`

5.8.2.6 `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.

5.8.2.7 `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`

5.9 Tasks

Functions

- `OPTIXAPI OptixResult optixTaskExecute (OptixTask task, OptixTask *additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int *numAdditionalTasksCreated)`
- `OPTIXAPI OptixResult optixTaskGetSerializationKey (OptixTask task, void *key, size_t *size)`
- `OPTIXAPI OptixResult optixTaskSerializeOutput (OptixTask task, void *data, size_t *size)`

- `OPTIXAPI OptixResult optixTaskDeserializeOutput` (`OptixTask` task, `const void *data`, `size_t size`, `OptixTask *additionalTasks`, `unsigned int maxNumAdditionalTasks`, `unsigned int *numAdditionalTasksCreated`)

5.9.1 Detailed Description

5.9.2 Function Documentation

5.9.2.1 `optixTaskDeserializeOutput()`

```
OPTIXAPI OptixResult optixTaskDeserializeOutput (
    OptixTask task,
    const void * data,
    size_t size,
    OptixTask * additionalTasks,
    unsigned int maxNumAdditionalTasks,
    unsigned int * numAdditionalTasksCreated )
```

Given the serialized task output, deserialize it and return potential new dependent tasks similar to `optixTaskExecute()`. Calling `optixTaskDeserializeOutput()` on a completed (either executed or deserialized) task will return an error.

Parameters

in	<i>task</i>	the <code>OptixTask</code> which to be deserialized
in	<i>data</i>	the deserialized task's output
in	<i>size</i>	the size of the deserialized task's output
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>

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

5.9.2.3 optixTaskGetSerializationKey()

```

OPTIXAPI OptixResult optixTaskGetSerializationKey (
    OptixTask task,
    void * key,
    size_t * size )

```

Retrieve the task's serialization key and its size. It is expected to call this function twice. Once to get the size and once to retrieve the key after space for it has been allocated. If the size of the key will be zero, the task will not be serializable and the task should be executed through [optixTaskExecute\(\)](#).

Parameters

in	<i>task</i>	the OptixTask which key to retrieve
out	<i>key</i>	characters representing the key without string-terminating '\0'. If nullptr, no output will be written
out	<i>size</i>	size of the key. Will be 0 for non-serializable tasks.

Returns

success

5.9.2.4 optixTaskSerializeOutput()

```

OPTIXAPI OptixResult optixTaskSerializeOutput (
    OptixTask task,
    void * data,
    size_t * size )

```

Retrieve the serialized data of the task's output. It is expected to call this function twice. Once to get the size and once to retrieve the data after space for it has been allocated. Calling [optixTaskSerializeOutput\(\)](#) before calling [optixTaskExecute\(\)](#) will return an error. Calling [optixTaskSerializeOutput\(\)](#) after calling [optixTaskDeserializeOutput\(\)](#) will return an error.

Parameters

in	<i>task</i>	the OptixTask which output data to retrieve
out	<i>data</i>	allocated space big enough to hold the output. If nullptr, no output will be written
out	<i>size</i>	size of the data. Will be 0 for non-serializable tasks.

5.10 Program groups

Functions

- `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 optixSbtRecordPackHeader (OptixProgramGroup programGroup, void *sbtRecordHeaderHostPointer)`

5.10.1 Detailed Description

5.10.2 Function Documentation

5.10.2.1 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	context	
in	programDescriptions	N * OptixProgramGroupDesc
in	numProgramGroups	N
in	options	
out	logString	Information will be written to this string. If logStringSize > 0 logString will be null terminated.
in, out	logStringSize	

Parameters

out	<i>programGroups</i>	
-----	----------------------	--

5.10.2.2 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.

5.10.2.3 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	<i>programGroup</i>	the program group
out	<i>stackSizes</i>	the corresponding stack sizes
in	<i>pipeline</i>	considering the program group within the given pipeline, can be <code>NULL</code>

5.10.2.4 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

5.11 Launches

Functions

- **OPTIXAPI OptixResult optixLaunch** (`OptixPipeline` pipeline, `CUstream` stream, `CUdeviceptr` pipelineParams, `size_t` pipelineParamsSize, `const OptixShaderBindingTable *sbt`, unsigned int width, unsigned int height, unsigned int depth)

5.11.1 Detailed Description

5.11.2 Function Documentation

5.11.2.1 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.

5.12 Acceleration structures

Functions

- `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 optixClusterAccelComputeMemoryUsage` (`OptixDeviceContext context`, `OptixClusterAccelBuildMode buildMode`, `const OptixClusterAccelBuildInput *buildInput`, `OptixAccelBufferSizes *bufferSizes`)
- OPTIXAPI `OptixResult optixClusterAccelBuild` (`OptixDeviceContext context`, `CUstream stream`, `const OptixClusterAccelBuildModeDesc *buildModeDesc`, `const OptixClusterAccelBuildInput *buildInput`, `CUdeviceptr argsArray`, `CUdeviceptr argsCount`, `unsigned int argsStrideInBytes`)

5.12.1 Detailed Description

5.12.2 Function Documentation

5.12.2.1 `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 OptixBuildInput objects
in	<i>numBuildInputs</i>	must be ≥ 1 for GAS, and $= 1$ for IAS
in	<i>tempBuffer</i>	must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT
in	<i>tempBufferSizeInBytes</i>	
in	<i>outputBuffer</i>	must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT
in	<i>outputBufferSizeInBytes</i>	
out	<i>outputHandle</i>	
in	<i>emittedProperties</i>	types of requested properties and output buffers
in	<i>numEmittedProperties</i>	number of post-build properties to populate (may be zero)

5.12.2.2 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>
in	<i>inputHandle</i>
in	<i>outputBuffer</i>

Parameters

in	<i>outputBufferSizeInBytes</i>
out	<i>outputHandle</i>

5.12.2.3 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 OptixBuildInput objects
in	<i>numBuildInputs</i>	number of elements in buildInputs (must be at least 1)
out	<i>bufferSizes</i>	fills in buffer sizes

5.12.2.4 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

5.12.2.5 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.

5.12.2.6 `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>

5.12.2.7 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>	
out	<i>compatible</i>	If OPTIX_SUCCESS is returned 'compatible' will have the value of either: • 0: This context is not compatible with the optix data structure associated with 'info'. • 1: This context is compatible.

5.12.2.8 optixClusterAccelBuild()

```

OPTIXAPI OptixResult optixClusterAccelBuild (
    OptixDeviceContext context,
    CUstream stream,
    const OptixClusterAccelBuildModeDesc * buildModeDesc,
    const OptixClusterAccelBuildInput * buildInput,
    CUdeviceptr argsArray,

```

```
CUdeviceptr argsCount,
unsigned int argsStrideInBytes )
```

Entry point to building one type of cluster objects: a CLAS, a Cluster template, or a GAS-over-CLAS. This is an indirect build function: all build arguments are read from device memory, with only the output location, type of build and limits passed on the host. This is a multi build function: more than one object can be built at once, but only of one type. The supplied limits must bound the inputs (Args) of all builds. Output buffer size constraints for implicit and explicit builds: implicit: The output and temp buffer must be at least as big as reported by a corresponding `optixClusterAccelComputeMemoryUsage` call. explicit: The output buffers must be at least as big as reported by a corresponding `optixClusterAccelBuild` call with the `getSize` mode and all device data supplied. The temp buffer must be at least as big as reported by a corresponding `optixClusterAccelComputeMemoryUsage` call. `getSize`: No output buffer is used. The temp buffer must be at least as big as reported by a corresponding `optixClusterAccelComputeMemoryUsage` call. Consequently, calling `optixClusterAccelBuild` with the `getSize` mode and subsequently building with the explicit mode is more memory efficient, but slower compared to building with the implicit mode.

Parameters

in	<i>context</i>	
in	<i>stream</i>	
in	<i>buildModeDesc</i>	A single input, describes where to write data for the selected build mode
in	<i>buildInput</i>	A single input, describes the type of object to build and limits over all objects' arguments
in	<i>argsArray</i>	Pointer to arguments array in device memory, describes each object to build: <code>OptixClusterAccelBuildInputTrianglesArgs</code> when using <code>OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES</code> <code>OptixClusterAccelBuildInputTrianglesArgs</code> when using <code>OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES</code> <code>OptixClusterAccelBuildInputGridsArgs</code> when using <code>OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS</code> <code>OptixClusterAccelBuildInputTemplatesArgs</code> when using <code>OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES</code> <code>OptixClusterAccelBuildInputClustersArgs</code> when using <code>OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS</code>
in	<i>argsCount</i>	Optional pointer to device memory, storing the number of objects to build, if null is provided, uses <code>maxArgCount</code> from <code>buildInput</code>
in	<i>argsStrideInBytes</i>	Optional stride of args objects, if null is provided, uses natural stride of Args type

5.12.2.9 `optixClusterAccelComputeMemoryUsage()`

```
OPTIXAPI OptixResult optixClusterAccelComputeMemoryUsage (
    OptixDeviceContext context,
    OptixClusterAccelBuildMode buildMode,
    const OptixClusterAccelBuildInput * buildInput,
    OptixAccelBufferSizes * bufferSizes )
```

Host side conservative memory computation for a subsequent `optixClusterAccelBuild` call with the same build mode and input. For implicit builds, the output buffer size contains the required size for holding all build outputs as specified in `buildInput->maxArgsCount`. For explicit builds, the output

buffer size contains the required size for holding a single build output. The temp buffer of any `optixClusterAccelBuild` call must be at least as big as reported by `optixClusterAccelComputeMemoryUsage`. `optixClusterAccelComputeMemoryUsage` always returns 0 for `OptixAccelBufferSizes::tempUpdateSizeInBytes`.

Parameters

in	<i>context</i>	
in	<i>buildMode</i>	Select the kind of output target (implicit: single buffer, explicit: per-build buffers, getSize: compact size computation for future explicit builds)
in	<i>buildInput</i>	A single input, describes the type of object to build and limits over all objects' arguments
out	<i>bufferSizes</i>	

5.12.2.10 `optixConvertPointerToTraversableHandle()`

```

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

```

Parameters

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

5.12.2.11 `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

5.12.2.12 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>

5.12.2.13 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>

Parameters

out	info
-----	------

5.12.2.14 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	context
in	stream
in	info
in	targetOpacityMicromapArray
in	targetOpacityMicromapArraySizeInBytes

5.13 Cooperative Vector

Functions

- [OPTIXAPI OptixResult optixCoopVecMatrixConvert](#) (OptixDeviceContext context, CUstream stream, unsigned int numNetworks, const [OptixNetworkDescription](#) *inputNetworkDescription, CUdeviceptr inputNetworks, size_t inputNetworkStrideInBytes, const [OptixNetworkDescription](#) *outputNetworkDescription, CUdeviceptr outputNetworks, size_t outputNetworkStrideInBytes)
- [OPTIXAPI OptixResult optixCoopVecMatrixComputeSize](#) (OptixDeviceContext context, unsigned int N, unsigned int K, [OptixCoopVecElemType](#) elementType,

`OptixCoopVecMatrixLayout` layout, `size_t` rowColumnStrideInBytes, `size_t` *sizeInBytes)

5.13.1 Detailed Description

5.13.2 Function Documentation

5.13.2.1 optixCoopVecMatrixComputeSize()

```
OPTIXAPI OptixResult optixCoopVecMatrixComputeSize (
    OptixDeviceContext context,
    unsigned int N,
    unsigned int K,
    OptixCoopVecElemType elementType,
    OptixCoopVecMatrixLayout layout,
    size_t rowColumnStrideInBytes,
    size_t * sizeInBytes )
```

For row and column ordered matrix layouts, when *rowColumnStrideInBytes* is 0, the stride will assume tight packing.

Results will be rounded to the next multiple of 64 to make it easy to pack the matrices in memory and have the correct alignment.

Parameters

in	<i>context</i>	
in	<i>elementType</i>	
in	<i>N</i>	
in	<i>K</i>	
in	<i>layout</i>	
in	<i>rowColumnStrideInBytes</i>	Ignored for optimal layouts
out	<i>sizeInBytes</i>	Output size of the matrix in bytes

5.13.2.2 optixCoopVecMatrixConvert()

```
OPTIXAPI OptixResult optixCoopVecMatrixConvert (
    OptixDeviceContext context,
    CUstream stream,
    unsigned int numNetworks,
    const OptixNetworkDescription * inputNetworkDescription,
    CUdeviceptr inputNetworks,
    size_t inputNetworkStrideInBytes,
    const OptixNetworkDescription * outputNetworkDescription,
    CUdeviceptr outputNetworks,
    size_t outputNetworkStrideInBytes )
```

Convert matrices from one layout and or element type to another.

One use case is to convert a matrix in `OPTIX_COOP_VEC_MATRIX_LAYOUT_ROW_MAJOR` or

OPTIX_COOP_VEC_MATRIX_LAYOUT_COLUMN_MAJOR into OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL.

The alignment base address + offset of each matrix needs to be a minimum of 64 bytes. This is similar to the requirements of [optixCoopVecMatMul](#).

Type conversion is possible, but is limited. If the input elementType and output elementType are not equal, then one must be OPTIX_COOP_VEC_ELEM_TYPE_FLOAT32 or OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16 and the other must be a lower-precision floating-point type. If the output elementType is OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E4M3 or OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E5M2, then the output layout must be OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL or OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL.

Parameters

in	<i>context</i>	
in	<i>stream</i>	
in	<i>numNetworks</i>	number of networks to convert
in	<i>inputNetworkDescription</i>	description of the input network matrix topology (one per invocation)
in	<i>inputNetworks</i>	base pointer to array of matrices that match the input topology specified in network
in	<i>inputNetworkStrideInBytes</i>	number of bytes between input networks, ignored if numNetworks is one
in	<i>outputNetworkDescription</i>	description of the output network matrix topology (one per invocation)
in	<i>outputNetworks</i>	base pointer to array of matrices that match the output topology specified in network
in	<i>outputNetworkStrideInBytes</i>	number of bytes between output networks, ignored if numNetworks is one

5.14 Denoiser

Functions

- 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)

5.14.1 Detailed Description

5.14.2 Function Documentation

5.14.2.1 `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

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

5.14.2.2 `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: $\text{intensity} = \log(r * 0.212586f + g * 0.715170f + b * 0.072200f)$. 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

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>	

5.14.2.3 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>

5.14.2.4 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>

5.14.2.5 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>
in	<i>userData</i>
in	<i>userDataSizeInBytes</i>
out	<i>denoiser</i>

5.14.2.6 optixDenoiserDestroy()

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

Destroys the denoiser object and any associated host resources.

5.14.2.7 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>

5.14.2.8 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>

5.15 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)

5.15.1 Detailed Description

OptiX Utilities.

5.15.2 Macro Definition Documentation

5.15.2.1 OPTIX_MICROMAP_FLOAT2_SUB

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

5.15.2.2 OPTIX_MICROMAP_INLINE_FUNC

```
#define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline
```

5.15.3 Function Documentation

5.15.3.1 __uint_as_float()

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

5.15.3.2 base2micro()

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

5.15.3.3 extractEvenBits()

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

5.15.3.4 index2dbary()

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

5.15.3.5 micro2bary()

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

5.15.3.6 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.

5.15.3.7 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](#)

5.15.3.8 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](#)

5.15.3.9 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.

5.15.3.10 `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.

5.15.3.11 `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.

5.15.3.12 `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.
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.

Parameters

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.

5.15.3.13 `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 maximas of the stack size requirements per array.

See programming guide for an explanation of the formula.

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

5.15.3.14 `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>

5.15.3.15 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

5.15.3.16 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

5.15.3.17 prefixEor()

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

5.16 Types

Classes

- struct [OptixDeviceContextOptions](#)
- struct [OptixOpacityMicromapUsageCount](#)
- struct [OptixBuildInputOpacityMicromap](#)
- struct [OptixRelocateInputOpacityMicromap](#)
- 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 OptixClusterAccelBuildModeDescImplicitDest
- struct OptixClusterAccelBuildModeDescExplicitDest
- struct OptixClusterAccelBuildModeDescGetSize
- struct OptixClusterAccelBuildInputTriangles
- struct OptixClusterAccelBuildInputGrids
- struct OptixClusterAccelBuildInputClusters
- struct OptixClusterAccelPrimitiveInfo
- struct OptixClusterAccelBuildInputTrianglesArgs
- struct OptixClusterAccelBuildInputGridsArgs
- struct OptixClusterAccelBuildInputTemplatesArgs
- struct OptixClusterAccelBuildInputClustersArgs
- struct OptixClusterAccelBuildInput
- struct OptixClusterAccelBuildModeDesc
- struct OptixImage2D
- struct OptixDenoiserOptions
- struct OptixDenoiserGuideLayer
- struct OptixDenoiserLayer
- struct OptixDenoiserParams
- struct OptixDenoiserSizes
- struct OptixTraverseData
- struct OptixModuleCompileBoundValueEntry
- struct OptixPayloadType
- struct OptixModuleCompileOptions
- struct OptixBuiltinISOOptions
- struct OptixProgramGroupSingleModule
- struct OptixProgramGroupHitgroup
- struct OptixProgramGroupCallables
- struct OptixProgramGroupDesc
- struct OptixProgramGroupOptions
- struct OptixPipelineCompileOptions
- struct OptixPipelineLinkOptions
- struct OptixShaderBindingTable
- struct OptixStackSizes
- struct OptixCoopVecMatrixDescription
- struct OptixNetworkDescription

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_PREDEFINED_INDEX_CLUSTER_SKIP_OPACITY_MICROMAP (-5)`
- `#define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull`
- `#define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12`

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 OptixPipelineSymbolMemcpyKind OptixPipelineSymbolMemcpyKind`
- `typedef enum OptixDevicePropertyShaderExecutionReorderingFlags OptixDevicePropertyShaderExecutionReorderingFlags`
- `typedef enum OptixDevicePropertyClusterAccelFlags OptixDevicePropertyClusterAccelFlags`
- `typedef enum OptixGeometryFlags OptixGeometryFlags`
- `typedef enum OptixHitKind OptixHitKind`
- `typedef enum OptixIndicesFormat OptixIndicesFormat`

- typedef enum OptixVertexFormat OptixVertexFormat
- typedef enum OptixTransformFormat OptixTransformFormat
- typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat
- typedef enum OptixOpacityMicromapArrayIndexingMode OptixOpacityMicromapArrayIndexingMode
- typedef struct OptixOpacityMicromapUsageCount OptixOpacityMicromapUsageCount
- typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap
- typedef struct OptixRelocateInputOpacityMicromap OptixRelocateInputOpacityMicromap
- 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 OptixClusterAccelBuildFlags OptixClusterAccelBuildFlags
- typedef enum OptixClusterAccelClusterFlags OptixClusterAccelClusterFlags
- typedef enum OptixClusterAccelPrimitiveFlags OptixClusterAccelPrimitiveFlags
- typedef enum OptixClusterAccelBuildType OptixClusterAccelBuildType
- typedef enum OptixClusterAccelBuildMode OptixClusterAccelBuildMode
- typedef enum OptixClusterAccelIndicesFormat OptixClusterAccelIndicesFormat

- typedef struct OptixClusterAccelBuildModeDescImplicitDest
OptixClusterAccelBuildModeDescImplicitDest
- typedef struct OptixClusterAccelBuildModeDescExplicitDest
OptixClusterAccelBuildModeDescExplicitDest
- typedef struct OptixClusterAccelBuildModeDescGetSize
OptixClusterAccelBuildModeDescGetSize
- typedef struct OptixClusterAccelBuildInputTriangles OptixClusterAccelBuildInputTriangles
- typedef struct OptixClusterAccelBuildInputGrids OptixClusterAccelBuildInputGrids
- typedef struct OptixClusterAccelBuildInputClusters OptixClusterAccelBuildInputClusters
- typedef struct OptixClusterAccelPrimitiveInfo OptixClusterAccelPrimitiveInfo
- typedef enum OptixClusterIDValues OptixClusterIDValues
- typedef struct OptixClusterAccelBuildInputTrianglesArgs
OptixClusterAccelBuildInputTrianglesArgs
- typedef struct OptixClusterAccelBuildInputGridsArgs OptixClusterAccelBuildInputGridsArgs
- typedef struct OptixClusterAccelBuildInputTemplatesArgs
OptixClusterAccelBuildInputTemplatesArgs
- typedef struct OptixClusterAccelBuildInputClustersArgs
OptixClusterAccelBuildInputClustersArgs
- typedef struct OptixClusterAccelBuildInput OptixClusterAccelBuildInput
- typedef struct OptixClusterAccelBuildModeDesc OptixClusterAccelBuildModeDesc
- 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 struct OptixTraverseData OptixTraverseData
- typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags
- typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
- typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
- typedef enum OptixModuleCompileState OptixModuleCompileState
- typedef enum OptixCreationFlags OptixCreationFlags
- typedef struct OptixModuleCompileBoundValueEntry OptixModuleCompileBoundValueEntry
- typedef enum OptixPayloadTypeID OptixPayloadTypeID
- typedef enum OptixPayloadSemantics OptixPayloadSemantics
- typedef struct OptixPayloadType OptixPayloadType
- typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
- typedef struct OptixBuiltinISOOptions OptixBuiltinISOOptions
- 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 OptixDevicePropertyCoopVecFlags OptixDevicePropertyCoopVecFlags
- typedef enum OptixCoopVecElemType OptixCoopVecElemType
- typedef enum OptixCoopVecMatrixLayout OptixCoopVecMatrixLayout
- typedef struct OptixCoopVecMatrixDescription OptixCoopVecMatrixDescription
- typedef struct OptixNetworkDescription OptixNetworkDescription
- typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
- typedef OptixResult() OptixQueryFunctionTable_t(int abiId, unsigned int numOptions, OptixQueryFunctionTableOptions *, const void **, void *functionTable, size_t sizeOfTable)

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_CREATION_CANCELED = 7290 ,
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_INVALID_POINTER = 7808 ,
OPTIX_ERROR_SYMBOL_NOT_FOUND = 7809 ,
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 ,
    OPTIX_DEVICE_PROPERTY_COOP_VEC = 0x200B ,
    OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL = 0x2020 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_VERTICES = 0x2021 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_TRIANGLES = 0x2022 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_STRUCTURED_GRID_RESOLUTION = 0x2023 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_SBT_INDEX = 0x2024 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTERS_PER_GAS = 0x2025 }

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

• enum OptixPipelineSymbolMemcpyKind {
    OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_FROM_DEVICE = 0x21A0 ,
    OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_FROM_HOST = 0x21A1 ,
    OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_TO_DEVICE = 0x21A2 ,
    OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_TO_HOST = 0x21A3 }

• enum OptixDevicePropertyShaderExecutionReorderingFlags {
    OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_NONE = 0 ,
    OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD = 1
    << 0 }

• enum OptixDevicePropertyClusterAccelFlags {
    OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_NONE = 0 ,
    OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_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 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 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_ROUND_QUADRATIC_BSPLINE_ROCAPS = 0x2508 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE_ROCAPS = 0x2509 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM_ROCAPS = 0x250A ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER_ROCAPS = 0x250B ,
OPTIX_PRIMITIVE_TYPE_TRIANGLE = 0x2531 }

```
  - 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_ROUND_QUADRATIC_BSPLINE_ROCAPS = 1 << 8 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE_ROCAPS = 1 << 9 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM_ROCAPS = 1 << 10 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER_ROCAPS = 1 << 11 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE = 1 << 31 }

```
  - 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 OptixClusterAccelBuildFlags {  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_FLAG\_NONE = 0 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_FLAG\_PREFER\_FAST\_TRACE = 1 << 0 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_FLAG\_PREFER\_FAST\_BUILD = 1 << 1 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_FLAG\_ALLOW\_OPACITY\_MICROMAPS = 1 << 2 }
- enum OptixClusterAccelClusterFlags {  
OPTIX\_CLUSTER\_ACCEL\_CLUSTER\_FLAG\_NONE = 0 ,  
OPTIX\_CLUSTER\_ACCEL\_CLUSTER\_FLAG\_ALLOW\_DISABLE\_OPACITY\_MICROMAPS = 1  
<< 0 }

- enum OptixClusterAccelPrimitiveFlags {  
OPTIX\_CLUSTER\_ACCEL\_PRIMITIVE\_FLAG\_NONE = 0 ,  
OPTIX\_CLUSTER\_ACCEL\_PRIMITIVE\_FLAG\_DISABLE\_TRIANGLE\_FACE\_CULLING = 1 << 0 ,  
OPTIX\_CLUSTER\_ACCEL\_PRIMITIVE\_FLAG\_REQUIRE\_SINGLE\_ANYHIT\_CALL = 1 << 1 ,  
OPTIX\_CLUSTER\_ACCEL\_PRIMITIVE\_FLAG\_DISABLE\_ANYHIT = 1 << 2 }
- enum OptixClusterAccelBuildType {  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_GASES\_FROM\_CLUSTERS = 0x2545 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_CLUSTERS\_FROM\_TRIANGLES = 0x2546 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_TEMPLATES\_FROM\_TRIANGLES = 0x2547 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_CLUSTERS\_FROM\_TEMPLATES = 0x2548 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_TEMPLATES\_FROM\_GRIDS = 0x2549 }
- enum OptixClusterAccelBuildMode {  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_MODE\_IMPLICIT\_DESTINATIONS = 0 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_MODE\_EXPLICIT\_DESTINATIONS = 1 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_MODE\_GET\_SIZES = 2 }
- enum OptixClusterAccelIndicesFormat {  
OPTIX\_CLUSTER\_ACCEL\_INDICES\_FORMAT\_8BIT = 1 ,  
OPTIX\_CLUSTER\_ACCEL\_INDICES\_FORMAT\_16BIT = 2 ,  
OPTIX\_CLUSTER\_ACCEL\_INDICES\_FORMAT\_32BIT = 4 }
- enum OptixClusterIDValues { OPTIX\_CLUSTER\_ID\_INVALID = 0xFFFFFFFFu }
- 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\_AOV = 0x2324 ,  
OPTIX\_DENOISER\_MODEL\_KIND\_TEMPORAL\_AOV = 0x2326 ,  
OPTIX\_DENOISER\_MODEL\_KIND\_UPSCALE2X = 0x2327 ,  
OPTIX\_DENOISER\_MODEL\_KIND\_TEMPORAL\_UPSCALE2X = 0x2328 ,  
OPTIX\_DENOISER\_MODEL\_KIND\_LDR = 0x2322 ,  
OPTIX\_DENOISER\_MODEL\_KIND\_HDR = 0x2323 ,  
OPTIX\_DENOISER\_MODEL\_KIND\_TEMPORAL = 0x2325 }
- 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 OptixCreationFlags {
 OPTIX_CREATION_FLAG_NONE = 0,
 OPTIX_CREATION_FLAG_BLOCK_UNTIL_EFFECTIVE = 1 << 0 }

• 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 OptixDevicePropertyCoopVecFlags {
 OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_NONE = 0 ,
 OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_STANDARD = 1 << 0 }

• enum OptixCoopVecElemType {
 OPTIX_COOP_VEC_ELEM_TYPE_UNKNOWN = 0x2A00 ,
 OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16 = 0x2A01 ,
 OPTIX_COOP_VEC_ELEM_TYPE_FLOAT32 = 0x2A03 ,
 OPTIX_COOP_VEC_ELEM_TYPE_UINT8 = 0x2A04 ,
 OPTIX_COOP_VEC_ELEM_TYPE_INT8 = 0x2A05 ,
 OPTIX_COOP_VEC_ELEM_TYPE_UINT32 = 0x2A08 ,
 OPTIX_COOP_VEC_ELEM_TYPE_INT32 = 0x2A09 ,
 OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E4M3 = 0x2A0A ,
 OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E5M2 = 0x2A0B }

• enum OptixCoopVecMatrixLayout {
 OPTIX_COOP_VEC_MATRIX_LAYOUT_ROW_MAJOR = 0x2A40 ,
 OPTIX_COOP_VEC_MATRIX_LAYOUT_COLUMN_MAJOR = 0x2A41 ,
 OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL = 0x2A42 ,
 OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL = 0x2A43 }

• enum OptixQueryFunctionTableOptions { OPTIX_QUERY_FUNCTION_TABLE_OPTION_
 DUMMY = 0 }

```

### 5.16.1 Detailed Description

OptiX Types.

### 5.16.2 Macro Definition Documentation

#### 5.16.2.1 OPTIX\_AABB\_BUFFER\_BYTE\_ALIGNMENT

```
#define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8u11
```

Alignment requirement for [OptixBuildInputCustomPrimitiveArray::aabbBuffers](#).

#### 5.16.2.2 OPTIX\_ACCEL\_BUFFER\_BYTE\_ALIGNMENT

```
#define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128u11
```

Alignment requirement for output and temporary buffers for acceleration structures.

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

```
#define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
```

Maximum number of payload types allowed.

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

```
#define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
```

Maximum number of payload values allowed.

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

```
#define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
```

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

#### 5.16.2.6 OPTIX\_GEOMETRY\_TRANSFORM\_BYTE\_ALIGNMENT

```
#define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16u11
```

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

#### 5.16.2.7 OPTIX\_INSTANCE\_BYTE\_ALIGNMENT

```
#define OPTIX_INSTANCE_BYTE_ALIGNMENT 16u11
```

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

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

```
#define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128u11
```

Alignment requirement for opacity micromap array buffers.

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

```
#define OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8u11
```

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

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

```
#define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
```

Maximum subdivision level for opacity micromaps.

## 5.16.2.11 OPTIX\_OPACITY\_MICROMAP\_PREDEFINED\_INDEX\_CLUSTER\_SKIP\_OPACITY\_MICROMAP

```
#define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_CLUSTER_SKIP_OPACITY_MICROMAP (-5)
```

Predefined index to indicate that no opacity micromap applies for a triangle. The opaque/non-opaque state is determined by the geometry flags, similar as for triangles in instances with the OPTIX\_INSTANCE\_FLAG\_DISABLE\_OPACITY\_MICROMAPS flag set. This special index is only available for the opacity micromap index array supplied to [OptixClusterAccelBuildInputTrianglesArgs](#). This special index does NOT require the cluster to be built with OPTIX\_CLUSTER\_ACCEL\_CLUSTER\_FLAG\_ALLOW\_DISABLE\_OPACITY\_MICROMAPS.

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

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

## 5.16.2.13 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.

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

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

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

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

## 5.16.2.16 OPTIX\_OPACITY\_MICROMAP\_STATE\_OPAQUE

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

## 5.16.2.17 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.

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

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

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

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

#### 5.16.2.20 OPTIX\_SBT\_RECORD\_ALIGNMENT

```
#define OPTIX_SBT_RECORD_ALIGNMENT 16ull
```

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

#### 5.16.2.21 OPTIX\_SBT\_RECORD\_HEADER\_SIZE

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

Size of the SBT record headers.

#### 5.16.2.22 OPTIX\_TRANSFORM\_BYTE\_ALIGNMENT

```
#define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
```

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

### 5.16.3 Typedef Documentation

#### 5.16.3.1 CUdeviceptr

```
typedef unsigned long long CUdeviceptr
```

CUDA device pointer.

#### 5.16.3.2 OptixAabb

```
typedef struct OptixAabb OptixAabb
```

AABB inputs.

#### 5.16.3.3 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\(\)](#)

#### 5.16.3.4 OptixAccelBuildOptions

```
typedef struct OptixAccelBuildOptions OptixAccelBuildOptions
```

Build options for acceleration structures.

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

#### 5.16.3.5 OptixAccelEmitDesc

```
typedef struct OptixAccelEmitDesc OptixAccelEmitDesc
```

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

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

#### 5.16.3.6 OptixAccelPropertyType

```
typedef enum OptixAccelPropertyType OptixAccelPropertyType
```

Properties which can be emitted during acceleration structure build.

See also `OptixAccelEmitDesc::type`.

### 5.16.3.7 OptixBuildFlags

```
typedef enum OptixBuildFlags OptixBuildFlags
```

Builder Options.

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

### 5.16.3.8 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()`

### 5.16.3.9 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  $d$  ( $3=\text{cubic}$ ,  $2=\text{quadratic}$ ,  $1=\text{linear}$ ) is represented by  $N > d$  vertices and  $N$  width values, and comprises  $N - d$  segments. Each segment is defined by  $d+1$  consecutive vertices. Each curve may have a different number of vertices.

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

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

See also `OptixBuildInput::curveArray`

### 5.16.3.10 OptixBuildInputCustomPrimitiveArray

```
typedef struct OptixBuildInputCustomPrimitiveArray
OptixBuildInputCustomPrimitiveArray
```

Custom primitive inputs.

See also `OptixBuildInput::customPrimitiveArray`

### 5.16.3.11 OptixBuildInputInstanceArray

```
typedef struct OptixBuildInputInstanceArray OptixBuildInputInstanceArray
```

Instance and instance pointer inputs.

See also `OptixBuildInput::instanceArray`

#### 5.16.3.12 OptixBuildInputOpacityMicromap

```
typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap
```

#### 5.16.3.13 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](#)

#### 5.16.3.14 OptixBuildInputTriangleArray

```
typedef struct OptixBuildInputTriangleArray OptixBuildInputTriangleArray
```

Triangle inputs.

See also [OptixBuildInput::triangleArray](#)

#### 5.16.3.15 OptixBuildInputType

```
typedef enum OptixBuildInputType OptixBuildInputType
```

Enum to distinguish the different build input types.

See also [OptixBuildInput::type](#)

#### 5.16.3.16 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](#)

#### 5.16.3.17 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\(\)](#)

#### 5.16.3.18 OptixClusterAccelBuildFlags

```
typedef enum OptixClusterAccelBuildFlags OptixClusterAccelBuildFlags
```

Host-side flags for all types of cluster builds.

#### 5.16.3.19 OptixClusterAccelBuildInput

```
typedef struct OptixClusterAccelBuildInput OptixClusterAccelBuildInput
```

#### 5.16.3.20 OptixClusterAccelBuildInputClusters

```
typedef struct OptixClusterAccelBuildInputClusters
OptixClusterAccelBuildInputClusters
```

#### 5.16.3.21 OptixClusterAccelBuildInputClustersArgs

```
typedef struct OptixClusterAccelBuildInputClustersArgs
OptixClusterAccelBuildInputClustersArgs
```

Device data, args provided for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_GASES\_FROM\_CLUSTERS builds.

#### 5.16.3.22 OptixClusterAccelBuildInputGrids

```
typedef struct OptixClusterAccelBuildInputGrids
OptixClusterAccelBuildInputGrids
```

#### 5.16.3.23 OptixClusterAccelBuildInputGridsArgs

```
typedef struct OptixClusterAccelBuildInputGridsArgs
OptixClusterAccelBuildInputGridsArgs
```

Device data, args provided for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_TEMPLATES\_FROM\_GRID builds.

#### 5.16.3.24 OptixClusterAccelBuildInputTemplatesArgs

```
typedef struct OptixClusterAccelBuildInputTemplatesArgs
OptixClusterAccelBuildInputTemplatesArgs
```

Device data, args provided for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_CLUSTERS\_FROM\_TEMPLATES builds.

#### 5.16.3.25 OptixClusterAccelBuildInputTriangles

```
typedef struct OptixClusterAccelBuildInputTriangles
OptixClusterAccelBuildInputTriangles
```

#### 5.16.3.26 OptixClusterAccelBuildInputTrianglesArgs

```
typedef struct OptixClusterAccelBuildInputTrianglesArgs
OptixClusterAccelBuildInputTrianglesArgs
```

Device data, args provided for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_CLUSTERS\_FROM\_TRIANGLES builds and OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_TEMPLATES\_FROM\_TRIANGLES builds.

### 5.16.3.27 OptixClusterAccelBuildMode

```
typedef enum OptixClusterAccelBuildMode OptixClusterAccelBuildMode
```

Build mode for cluster builds.

### 5.16.3.28 OptixClusterAccelBuildModeDesc

```
typedef struct OptixClusterAccelBuildModeDesc OptixClusterAccelBuildModeDesc
```

### 5.16.3.29 OptixClusterAccelBuildModeDescExplicitDest

```
typedef struct OptixClusterAccelBuildModeDescExplicitDest
OptixClusterAccelBuildModeDescExplicitDest
```

### 5.16.3.30 OptixClusterAccelBuildModeDescGetSize

```
typedef struct OptixClusterAccelBuildModeDescGetSize
OptixClusterAccelBuildModeDescGetSize
```

### 5.16.3.31 OptixClusterAccelBuildModeDescImplicitDest

```
typedef struct OptixClusterAccelBuildModeDescImplicitDest
OptixClusterAccelBuildModeDescImplicitDest
```

### 5.16.3.32 OptixClusterAccelBuildType

```
typedef enum OptixClusterAccelBuildType OptixClusterAccelBuildType
```

Build type for cluster builds - specifying the type of data input and output.

### 5.16.3.33 OptixClusterAccelClusterFlags

```
typedef enum OptixClusterAccelClusterFlags OptixClusterAccelClusterFlags
```

Device-side flags for clusters builds.

### 5.16.3.34 OptixClusterAccelIndicesFormat

```
typedef enum OptixClusterAccelIndicesFormat OptixClusterAccelIndicesFormat
```

Helper enum where values match the byte count of the corresponding index format, allowing usage of enum value when specifying byte count.

### 5.16.3.35 OptixClusterAccelPrimitiveFlags

```
typedef enum OptixClusterAccelPrimitiveFlags OptixClusterAccelPrimitiveFlags
```

Device-side flags that specify per-primitive specific behavior Note the packing within the 32b struct `OptixClusterAccelPrimitiveInfo`.

### 5.16.3.36 OptixClusterAccelPrimitiveInfo

```
typedef struct OptixClusterAccelPrimitiveInfo OptixClusterAccelPrimitiveInfo
```

### 5.16.3.37 OptixClusterIDValues

```
typedef enum OptixClusterIDValues OptixClusterIDValues
```

Reserved value for cluster IDs in Args.

### 5.16.3.38 OptixCompileDebugLevel

```
typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
```

Debug levels.

See also `OptixModuleCompileOptions::debugLevel`

### 5.16.3.39 OptixCompileOptimizationLevel

```
typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
```

Optimization levels.

See also `OptixModuleCompileOptions::optLevel`

### 5.16.3.40 OptixCoopVecElemType

```
typedef enum OptixCoopVecElemType OptixCoopVecElemType
```

### 5.16.3.41 OptixCoopVecMatrixDescription

```
typedef struct OptixCoopVecMatrixDescription OptixCoopVecMatrixDescription
```

Each matrix's offset from the base address is expressed with `offsetInBytes`. This allows for non-uniform matrices to be tightly packed.

The `rowColumnStrideInBytes` is ignored if the layout is either `OPTIX_COOP_VEC_MATRIX_LAYOUT_INTERFERENCING_OPTIMAL` or `OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL`

### 5.16.3.42 OptixCoopVecMatrixLayout

```
typedef enum OptixCoopVecMatrixLayout OptixCoopVecMatrixLayout
```

### 5.16.3.43 OptixCreationFlags

```
typedef enum OptixCreationFlags OptixCreationFlags
```

Flags for canceling the creation of an OptiX object.

If `OPTIX_CREATION_FLAG_BLOCK_UNTIL_EFFECTIVE` is set, the calling thread will block until one of these conditions is met:

1. All executing object creation threads have processed the new state
2. The creation of the object has finished, in which case the new state will be ignored

If `OPTIX_CREATION_FLAG_BLOCK_UNTIL_EFFECTIVE` is not set, any *CancelCreation* call will return without blocking. Note that the cancel request may still be ignored if all creation threads finish their tasks before they can process the new state.

See also `optixModuleCancelCreation()`, `#optixPipelineCancelCreations()`, `optixDeviceContextCancelCreations()`

### 5.16.3.44 OptixCurveEndcapFlags

```
typedef enum OptixCurveEndcapFlags OptixCurveEndcapFlags
```

Curve end cap types, for non-linear curves.

#### 5.16.3.45 OptixDenoiser

```
typedef struct OptixDenoiser_t* OptixDenoiser
```

Opaque type representing a denoiser instance.

#### 5.16.3.46 OptixDenoiserAlphaMode

```
typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
```

Alpha denoising mode.

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

#### 5.16.3.47 OptixDenoiserAOVType

```
typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
```

AOV type used by the denoiser.

#### 5.16.3.48 OptixDenoiserGuideLayer

```
typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
```

Guide layer for the denoiser.

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

#### 5.16.3.49 OptixDenoiserLayer

```
typedef struct OptixDenoiserLayer OptixDenoiserLayer
```

Input/Output layers for the denoiser.

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

#### 5.16.3.50 OptixDenoiserModelKind

```
typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
```

Model kind used by the denoiser.

See also [optixDenoiserCreate](#)

#### 5.16.3.51 OptixDenoiserOptions

```
typedef struct OptixDenoiserOptions OptixDenoiserOptions
```

Options used by the denoiser.

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

#### 5.16.3.52 OptixDenoiserParams

```
typedef struct OptixDenoiserParams OptixDenoiserParams
```

Various parameters used by the denoiser.

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

[optixDenoiserComputeIntensity\(\)](#)

[optixDenoiserComputeAverageColor\(\)](#)

### 5.16.3.53 OptixDenoiserSizes

```
typedef struct OptixDenoiserSizes OptixDenoiserSizes
```

Various sizes related to the denoiser.

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

### 5.16.3.54 OptixDeviceContext

```
typedef struct OptixDeviceContext_t* OptixDeviceContext
```

Opaque type representing a device context.

### 5.16.3.55 OptixDeviceContextOptions

```
typedef struct OptixDeviceContextOptions OptixDeviceContextOptions
```

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

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

### 5.16.3.56 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\(\)](#)

### 5.16.3.57 OptixDeviceProperty

```
typedef enum OptixDeviceProperty OptixDeviceProperty
```

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

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

### 5.16.3.58 OptixDevicePropertyClusterAccelFlags

```
typedef enum OptixDevicePropertyClusterAccelFlags
OptixDevicePropertyClusterAccelFlags
```

Flags used to interpret the result of [optixDeviceContextGetProperty\(\)](#) and `OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL`.

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

### 5.16.3.59 OptixDevicePropertyCoopVecFlags

```
typedef enum OptixDevicePropertyCoopVecFlags OptixDevicePropertyCoopVecFlags
```

Flags used to interpret the result of [optixDeviceContextGetProperty\(\)](#) and `OPTIX_DEVICE_PROPERTY_COOP_VEC`.

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

### 5.16.3.60 OptixDevicePropertyShaderExecutionReorderingFlags

```
typedef enum OptixDevicePropertyShaderExecutionReorderingFlags
OptixDevicePropertyShaderExecutionReorderingFlags
```

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

See also `optixDeviceContextGetProperty()`

### 5.16.3.61 OptixExceptionCodes

```
typedef enum OptixExceptionCodes OptixExceptionCodes
```

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

### 5.16.3.62 OptixExceptionFlags

```
typedef enum OptixExceptionFlags OptixExceptionFlags
```

Exception flags.

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

### 5.16.3.63 OptixGeometryFlags

```
typedef enum OptixGeometryFlags OptixGeometryFlags
```

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

### 5.16.3.64 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()`

### 5.16.3.65 OptixImage2D

```
typedef struct OptixImage2D OptixImage2D
```

Image descriptor used by the denoiser.

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

### 5.16.3.66 OptixIndicesFormat

```
typedef enum OptixIndicesFormat OptixIndicesFormat
```

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

### 5.16.3.67 OptixInstance

```
typedef struct OptixInstance OptixInstance
```

Instances.

See also `OptixBuildInputInstanceArray::instances`

### 5.16.3.68 OptixInstanceFlags

```
typedef enum OptixInstanceFlags OptixInstanceFlags
```

Flags set on the `OptixInstance::flags`.

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

### 5.16.3.69 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`

### 5.16.3.70 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()`

### 5.16.3.71 OptixMicromapBuffers

```
typedef struct OptixMicromapBuffers OptixMicromapBuffers
```

Buffer inputs for opacity micromap array builds.

### 5.16.3.72 OptixMicromapBufferSizes

```
typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
```

Conservative memory requirements for building a opacity micromap array.

### 5.16.3.73 OptixModule

```
typedef struct OptixModule_t* OptixModule
```

Opaque type representing a module.

### 5.16.3.74 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](#)

### 5.16.3.75 OptixModuleCompileOptions

```
typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
```

Compilation options for module.

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

#### 5.16.3.76 OptixModuleCompileState

```
typedef enum OptixModuleCompileState OptixModuleCompileState
```

Module compilation state.

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

#### 5.16.3.77 OptixMotionFlags

```
typedef enum OptixMotionFlags OptixMotionFlags
```

Enum to specify motion flags.

See also `OptixMotionOptions::flags`.

#### 5.16.3.78 OptixMotionOptions

```
typedef struct OptixMotionOptions OptixMotionOptions
```

Motion options.

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

#### 5.16.3.79 OptixNetworkDescription

```
typedef struct OptixNetworkDescription OptixNetworkDescription
```

#### 5.16.3.80 OptixOpacityMicromapArrayBuildInput

```
typedef struct OptixOpacityMicromapArrayBuildInput
OptixOpacityMicromapArrayBuildInput
```

Inputs to opacity micromap array construction.

#### 5.16.3.81 OptixOpacityMicromapArrayIndexingMode

```
typedef enum OptixOpacityMicromapArrayIndexingMode
OptixOpacityMicromapArrayIndexingMode
```

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

#### 5.16.3.82 OptixOpacityMicromapDesc

```
typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc
```

Opacity micromap descriptor.

#### 5.16.3.83 OptixOpacityMicromapFlags

```
typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags
```

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

#### 5.16.3.84 OptixOpacityMicromapFormat

```
typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat
```

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

### 5.16.3.85 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.

### 5.16.3.86 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.

### 5.16.3.87 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.

### 5.16.3.88 OptixPayloadType

```
typedef struct OptixPayloadType OptixPayloadType
```

Specifies a single payload type.

### 5.16.3.89 OptixPayloadTypeID

```
typedef enum OptixPayloadTypeID OptixPayloadTypeID
```

Payload type identifiers.

### 5.16.3.90 OptixPipeline

```
typedef struct OptixPipeline_t* OptixPipeline
```

Opaque type representing a pipeline.

### 5.16.3.91 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()`

### 5.16.3.92 OptixPipelineLinkOptions

```
typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
```

Link options for a pipeline.

See also `optixPipelineCreate()`

### 5.16.3.93 OptixPipelineSymbolMemcpyKind

```
typedef enum OptixPipelineSymbolMemcpyKind OptixPipelineSymbolMemcpyKind
```

Flags used to interpret the source and target of memory copies when using `optixPipelineSymbolMemcpyAsync()`

See also `optixPipelineSymbolMemcpyAsync()`

### 5.16.3.94 OptixPixelFormat

```
typedef enum OptixPixelFormat OptixPixelFormat
```

Pixel formats used by the denoiser.

See also `OptixImage2D::format`

### 5.16.3.95 OptixPrimitiveType

```
typedef enum OptixPrimitiveType OptixPrimitiveType
```

Builtin primitive types.

### 5.16.3.96 OptixPrimitiveTypeFlags

```
typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags
```

Builtin flags may be bitwise combined.

See also `OptixPipelineCompileOptions::usesPrimitiveTypeFlags`

### 5.16.3.97 OptixProgramGroup

```
typedef struct OptixProgramGroup_t* OptixProgramGroup
```

Opaque type representing a program group.

### 5.16.3.98 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`

### 5.16.3.99 OptixProgramGroupDesc

```
typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
```

Descriptor for program groups.

### 5.16.3.100 OptixProgramGroupFlags

```
typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
```

Flags for program groups.

### 5.16.3.101 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](#)

### 5.16.3.102 OptixProgramGroupKind

```
typedef enum OptixProgramGroupKind OptixProgramGroupKind
```

Distinguishes different kinds of program groups.

### 5.16.3.103 OptixProgramGroupOptions

```
typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
```

Program group options.

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

### 5.16.3.104 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](#)

### 5.16.3.105 OptixQueryFunctionTable\_t

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

Type of the function [optixQueryFunctionTable\(\)](#)

### 5.16.3.106 OptixQueryFunctionTableOptions

```
typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
```

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

### 5.16.3.107 OptixRayFlags

```
typedef enum OptixRayFlags OptixRayFlags
```

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

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

#### 5.16.3.108 OptixRelocateInput

```
typedef struct OptixRelocateInput OptixRelocateInput
```

Relocation inputs.

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

#### 5.16.3.109 OptixRelocateInputInstanceArray

```
typedef struct OptixRelocateInputInstanceArray
OptixRelocateInputInstanceArray
```

Instance and instance pointer inputs.

See also [OptixRelocateInput::instanceArray](#)

#### 5.16.3.110 OptixRelocateInputOpacityMicromap

```
typedef struct OptixRelocateInputOpacityMicromap
OptixRelocateInputOpacityMicromap
```

#### 5.16.3.111 OptixRelocateInputTriangleArray

```
typedef struct OptixRelocateInputTriangleArray
OptixRelocateInputTriangleArray
```

Triangle inputs.

See also [OptixRelocateInput::triangleArray](#)

#### 5.16.3.112 OptixRelocationInfo

```
typedef struct OptixRelocationInfo OptixRelocationInfo
```

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

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

#### 5.16.3.113 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\(\)](#)

#### 5.16.3.114 OptixShaderBindingTable

```
typedef struct OptixShaderBindingTable OptixShaderBindingTable
```

Describes the shader binding table (SBT)

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

### 5.16.3.115 OptixSRTData

`typedef struct OptixSRTData OptixSRTData`

Represents an SRT transformation.

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

The scaling matrix  $S = \begin{bmatrix} sx & a & b & pvx \\ 0 & sy & c & pvy \\ 0 & 0 & sz & pvz \end{bmatrix}$  defines an affine transformation that can include scale,

shear, and a translation. The translation allows to define the pivot point for the subsequent rotation.

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

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

Typically, this translation includes the inverse translation from the matrix S to reverse the translation for the pivot point for R.

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

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

### 5.16.3.116 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\(\)](#)

### 5.16.3.117 OptixStackSizes

`typedef struct OptixStackSizes OptixStackSizes`

Describes the stack size requirements of a program group.

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

### 5.16.3.118 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\(\)](#)

### 5.16.3.119 OptixTask

```
typedef struct OptixTask_t* OptixTask
```

Opaque type representing a work task.

### 5.16.3.120 OptixTransformFormat

```
typedef enum OptixTransformFormat OptixTransformFormat
```

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

### 5.16.3.121 OptixTransformType

```
typedef enum OptixTransformType OptixTransformType
```

Transform.

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

### 5.16.3.122 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.

### 5.16.3.123 OptixTraversableHandle

```
typedef unsigned long long OptixTraversableHandle
```

Traversable handle.

### 5.16.3.124 OptixTraversableType

```
typedef enum OptixTraversableType OptixTraversableType
```

Traversable Handles.

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

### 5.16.3.125 OptixTraverseData

```
typedef struct OptixTraverseData OptixTraverseData
```

Hit Object Struct to store the data collected in a hit object during traversal in an internal format using [optixHitObjectGetTraverseData\(\)](#). The hit object can be reconstructed using that data at a later point with [optixMakeHitObjectWithTraverseData\(\)](#).

### 5.16.3.126 OptixVertexFormat

typedef enum [OptixVertexFormat](#) [OptixVertexFormat](#)

Format of vertices used in [OptixBuildInputTriangleArray::vertexFormat](#).

### 5.16.3.127 OptixVisibilityMask

typedef unsigned int [OptixVisibilityMask](#)

Visibility mask.

## 5.16.4 Enumeration Type Documentation

### 5.16.4.1 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          | <a href="#">OptixAabb</a> * numMotionSteps.                                        |

### 5.16.4.2 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 <a href="#">optixAccelBuild</a> . |
| OPTIX_BUILD_FLAG_ALLOW_COMPACTION  |                                                                                                               |
| OPTIX_BUILD_FLAG_PREFER_FAST_TRACE | This flag is mutually exclusive with OPTIX_BUILD_FLAG_PREFER_FAST_BUILD.                                      |
| OPTIX_BUILD_FLAG_PREFER_FAST_BUILD | This flag is mutually exclusive with OPTIX_BUILD_FLAG_PREFER_FAST_TRACE.                                      |

## Enumerator

|                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS      | Allow random access to build input vertices See<br><a href="#">optixGetTriangleVertexDataFromHandle</a><br><a href="#">optixGetLinearCurveVertexDataFromHandle</a><br><a href="#">optixGetQuadraticBSplineVertexDataFromHandle</a><br><a href="#">optixGetCubicBSplineVertexDataFromHandle</a><br><a href="#">optixGetCatmullRomVertexDataFromHandle</a><br><a href="#">optixGetCubicBezierVertexDataFromHandle</a><br><a href="#">optixGetQuadraticBSplineRocapsVertexDataFromHandle</a><br><a href="#">optixGetCubicBSplineRocapsVertexDataFromHandle</a><br><a href="#">optixGetCatmullRomRocapsVertexDataFromHandle</a><br><a href="#">optixGetCubicBezierRocapsVertexDataFromHandle</a><br><a href="#">optixGetRibbonVertexDataFromHandle</a><br><a href="#">optixGetRibbonNormalFromHandle</a><br><a href="#">optixGetSphereDataFromHandle</a> . |
| OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS    | Allow random access to instances See<br><a href="#">optixGetInstanceTraversableFromIAS</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 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 <code>DISABLE_OPACITY_MICROMAPS</code> flag instance flag. Note that the GAS will not be optimized for the attached opacity micromap Arrays if this flag is set, which may result in reduced traversal performance.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## 5.16.4.3 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<br><a href="#">OptixBuildInputTriangleArray</a>                |
| OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES | Custom primitive inputs. See also<br><a href="#">OptixBuildInputCustomPrimitiveArray</a> |

## Enumerator

|                                          |                                                                                |
|------------------------------------------|--------------------------------------------------------------------------------|
| 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>             |

## 5.16.4.4 OptixBuildOperation

enum [OptixBuildOperation](#)

Enum to specify the acceleration build operation.

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

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

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

## Enumerator

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

## 5.16.4.5 OptixClusterAccelBuildFlags

enum [OptixClusterAccelBuildFlags](#)

Host-side flags for all types of cluster builds.

## Enumerator

|                                                        |  |
|--------------------------------------------------------|--|
| OPTIX_CLUSTER_ACCEL_BUILD_FLAG_NONE                    |  |
| OPTIX_CLUSTER_ACCEL_BUILD_FLAG_PREFER_FAST_TRACE       |  |
| OPTIX_CLUSTER_ACCEL_BUILD_FLAG_PREFER_FAST_BUILD       |  |
| OPTIX_CLUSTER_ACCEL_BUILD_FLAG_ALLOW_OPACITY_MICROMAPS |  |

## 5.16.4.6 OptixClusterAccelBuildMode

enum [OptixClusterAccelBuildMode](#)

Build mode for cluster builds.

## Enumerator

|                                                      |                                                                                                           |
|------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| OPTIX_CLUSTER_ACCEL_BUILD_MODE_IMPLICIT_DESTINATIONS | Fastest build, single output buffer, build outputs may have padding wrt each other.                       |
| OPTIX_CLUSTER_ACCEL_BUILD_MODE_EXPLICIT_DESTINATIONS | Compact build, application specifies output destination per Arg; requires Get Sizes build run beforehand. |
| OPTIX_CLUSTER_ACCEL_BUILD_MODE_GET_SIZES             | Size computation for future explicit build; computes output sizes for all Args.                           |

## 5.16.4.7 OptixClusterAccelBuildType

enum `OptixClusterAccelBuildType`

Build type for cluster builds - specifying the type of data input and output.

## Enumerator

|                                                         |
|---------------------------------------------------------|
| OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS      |
| OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES  |
| OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES |
| OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES  |
| OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS     |

## 5.16.4.8 OptixClusterAccelClusterFlags

enum `OptixClusterAccelClusterFlags`

Device-side flags for clusters builds.

## Enumerator

|                                                                  |                                                                                                                                                                                                                  |
|------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPTIX_CLUSTER_ACCEL_CLUSTER_FLAG_NONE                            |                                                                                                                                                                                                                  |
| OPTIX_CLUSTER_ACCEL_CLUSTER_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS | Similar to the 'ALLOW_DISABLE_OPACITY_MICROMAPS' build flag of regular triangle GAS builds. This flag is required if the CLAS is in an instance with the OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS flag set. |

## 5.16.4.9 OptixClusterAccelIndicesFormat

enum `OptixClusterAccelIndicesFormat`

Helper enum where values match the byte count of the corresponding index format, allowing usage of enum value when specifying byte count.

## Enumerator

|                                          |
|------------------------------------------|
| OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_8BIT  |
| OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_16BIT |

Enumerator

|                                          |
|------------------------------------------|
| OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_32BIT |
|------------------------------------------|

#### 5.16.4.10 OptixClusterAccelPrimitiveFlags

enum [OptixClusterAccelPrimitiveFlags](#)

Device-side flags that specify per-primitive specific behavior Note the packing within the 32b struct [OptixClusterAccelPrimitiveInfo](#).

Enumerator

|                                                                  |
|------------------------------------------------------------------|
| OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_NONE                          |
| OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_TRIANGLE_FACE_CULLING |
| OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_REQUIRE_SINGLE_ANYHIT_CALL    |
| OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_ANYHIT                |

#### 5.16.4.11 OptixClusterIDValues

enum [OptixClusterIDValues](#)

Reserved value for cluster IDs in Args.

Enumerator

|                          |
|--------------------------|
| OPTIX_CLUSTER_ID_INVALID |
|--------------------------|

#### 5.16.4.12 OptixCompileDebugLevel

enum [OptixCompileDebugLevel](#)

Debug levels.

See also [OptixModuleCompileOptions::debugLevel](#)

Enumerator

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

#### 5.16.4.13 OptixCompileOptimizationLevel

enum [OptixCompileOptimizationLevel](#)

Optimization levels.

See also [OptixModuleCompileOptions::optLevel](#)

Enumerator

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

#### 5.16.4.14 OptixCoopVecElemType

enum [OptixCoopVecElemType](#)

Enumerator

|                                      |                                                                                                                                                |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| OPTIX_COOP_VEC_ELEM_TYPE_UNKNOWN     |                                                                                                                                                |
| OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16     | 16 bit float                                                                                                                                   |
| OPTIX_COOP_VEC_ELEM_TYPE_FLOAT32     | 32 bit float                                                                                                                                   |
| OPTIX_COOP_VEC_ELEM_TYPE_UINT8       | 8 bit unsigned integer                                                                                                                         |
| OPTIX_COOP_VEC_ELEM_TYPE_INT8        | 8 bit signed integer                                                                                                                           |
| OPTIX_COOP_VEC_ELEM_TYPE_UINT32      | 32 bit unsigned integer                                                                                                                        |
| OPTIX_COOP_VEC_ELEM_TYPE_INT32       | 32 bit signed integer                                                                                                                          |
| OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E4M3 | FLOAT8 type with 4 bits exponent, 3 bits mantissa. Only supported as the <code>inputInterpretation</code> and <code>matrixElementType</code> . |
| OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E5M2 | FLOAT8 type with 5 bits exponent, 2 bits mantissa. Only supported as the <code>inputInterpretation</code> and <code>matrixElementType</code> . |

#### 5.16.4.15 OptixCoopVecMatrixLayout

enum [OptixCoopVecMatrixLayout](#)

Enumerator

|                                                  |
|--------------------------------------------------|
| OPTIX_COOP_VEC_MATRIX_LAYOUT_ROW_MAJOR           |
| OPTIX_COOP_VEC_MATRIX_LAYOUT_COLUMN_MAJOR        |
| OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL |
| OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL    |

#### 5.16.4.16 OptixCreationFlags

enum [OptixCreationFlags](#)

Flags for canceling the creation of an OptiX object.

If `OPTIX_CREATION_FLAG_BLOCK_UNTIL_EFFECTIVE` is set, the calling thread will block until one of these conditions is met:

1. All executing object creation threads have processed the new state
2. The creation of the object has finished, in which case the new state will be ignored

If `OPTIX_CREATION_FLAG_BLOCK_UNTIL_EFFECTIVE` is not set, any *CancelCreation* call will return without blocking. Note that the cancel request may still be ignored if all creation threads finish their tasks before they can process the new state.

See also `optixModuleCancelCreation()`, `#optixPipelineCancelCreations()`, `optixDeviceContextCancelCreations()`

Enumerator

|                                                        |
|--------------------------------------------------------|
| <code>OPTIX_CREATION_FLAG_NONE</code>                  |
| <code>OPTIX_CREATION_FLAG_BLOCK_UNTIL_EFFECTIVE</code> |

#### 5.16.4.17 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.       |
| <code>OPTIX_CURVE_ENDCAP_ON</code>      | Flat end caps at both ends of quadratic/cubic curve segments. Not valid for linear. |

#### 5.16.4.18 OptixDenoiserAlphaMode

enum `OptixDenoiserAlphaMode`

Alpha denoising mode.

See also `optixDenoiserCreate()`

Enumerator

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

#### 5.16.4.19 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    |                       |

## 5.16.4.20 OptixDenoiserModelKind

enum [OptixDenoiserModelKind](#)

Model kind used by the denoiser.

See also [optixDenoiserCreate](#)

## Enumerator

|                                              |                                                                                                               |
|----------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| OPTIX_DENOISER_MODEL_KIND_AOV                | Built-in model for denoising single image.                                                                    |
| OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV       | Built-in model for denoising image sequence, temporally stable.                                               |
| OPTIX_DENOISER_MODEL_KIND_UPSCALE2X          | Built-in model for denoising single image upscaling (supports AOVs).                                          |
| OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X | Built-in model for denoising image sequence upscaling, temporally stable (supports AOVs).                     |
| OPTIX_DENOISER_MODEL_KIND_LDR                | Deprecated. Use OPTIX_DENOISER_MODEL_KIND_AOV. When used, internally mapped to OPTIX_DENOISER_MODEL_KIND_AOV. |
| OPTIX_DENOISER_MODEL_KIND_HDR                |                                                                                                               |
| OPTIX_DENOISER_MODEL_KIND_TEMPORAL           | Deprecated. Use OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV.                                                       |

## 5.16.4.21 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 |

### 5.16.4.22 OptixDeviceProperty

enum `OptixDeviceProperty`

Parameters used for `optixDeviceContextGetProperty()`

See also `optixDeviceContextGetProperty()`

Enumerator

|                                                                            |                                                                                                                                                                                                                                          |
|----------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH</code>                   | Maximum value for <code>OptixPipelineLinkOptions::maxTraceDepth</code> . sizeof(unsigned int)                                                                                                                                            |
| <code>OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH</code>       | Maximum value to pass into <code>optixPipelineSetStackSize</code> for parameter <code>maxTraversableGraphDepth</code> . sizeof(unsigned int)                                                                                             |
| <code>OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS</code>            | The maximum number of primitives (over all build inputs) as input to a single Geometry Acceleration Structure (GAS). sizeof(unsigned int)                                                                                                |
| <code>OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS</code>             | The maximum number of instances (over all build inputs) as input to a single Instance Acceleration Structure (IAS). sizeof(unsigned int)                                                                                                 |
| <code>OPTIX_DEVICE_PROPERTY_RTCORE_VERSION</code>                          | The RT core version supported by the device (0 for no support, 10 for version 1.0). sizeof(unsigned int)                                                                                                                                 |
| <code>OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID</code>                   | The maximum value for <code>OptixInstance::instanceId</code> . sizeof(unsigned int)                                                                                                                                                      |
| <code>OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK</code> | The number of bits available for the <code>OptixInstance::visibilityMask</code> . Higher bits must be set to zero. sizeof(unsigned int)                                                                                                  |
| <code>OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS</code>           | The maximum number of instances that can be added to a single Instance Acceleration Structure (IAS). sizeof(unsigned int)                                                                                                                |
| <code>OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET</code>                    | 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)                                         |
| <code>OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING</code>             | 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) |
| <code>OPTIX_DEVICE_PROPERTY_COOP_VEC</code>                                | Returns a flag specifying whether cooperative vector support is enabled for this device. See <code>OptixDevicePropertyCoopVecFlags</code> for documentation on the values that can be returned. sizeof(unsigned int)                     |

## Enumerator

|                                                            |                                                                                                                                                                                                                            |
|------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL                        | Returns a flag specifying support for cluster acceleration structure builds. See <code>OptixDevicePropertyClusterAccelFlags</code> for documentation on the values that can be returned. <code>sizeof(unsigned int)</code> |
| OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_VERTICES           | Returns a maximum unique vertices per cluster in a cluster acceleration structure (CLAS) build. <code>sizeof(unsigned int)</code>                                                                                          |
| OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_TRIANGLES          | Returns a maximum triangles per cluster in a cluster acceleration structure (CLAS) build. <code>sizeof(unsigned int)</code>                                                                                                |
| OPTIX_DEVICE_PROPERTY_LIMIT_MAX_STRUCTURED_GRID_RESOLUTION | Returns a maximum resolution per cluster in a structured cluster acceleration (CLAS) structure build. <code>sizeof(unsigned int)</code>                                                                                    |
| OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_SBT_INDEX          | Returns a maximum sbt index allowed in a cluster acceleration structure (CLAS) build. <code>sizeof(unsigned int)</code>                                                                                                    |
| OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTERS_PER_GAS           | Returns the maximum number of clusters (CLAS) as input to a single Geometry Acceleration Structure (GAS). <code>sizeof(unsigned int)</code>                                                                                |

5.16.4.23 `OptixDevicePropertyClusterAccelFlags`enum `OptixDevicePropertyClusterAccelFlags`

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

See also `optixDeviceContextGetProperty()`

## Enumerator

|                                                   |                                                          |
|---------------------------------------------------|----------------------------------------------------------|
| OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_NONE     | Cluster acceleration structure builds are not supported. |
| OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_STANDARD |                                                          |

5.16.4.24 `OptixDevicePropertyCoopVecFlags`enum `OptixDevicePropertyCoopVecFlags`

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

See also `optixDeviceContextGetProperty()`

## Enumerator

|                                          |                                                                                       |
|------------------------------------------|---------------------------------------------------------------------------------------|
| OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_NONE | Any use of cooperative vector host APIs or device intrinsics will result in an error. |
|------------------------------------------|---------------------------------------------------------------------------------------|

Enumerator

|                                              |  |
|----------------------------------------------|--|
| OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_STANDARD |  |
|----------------------------------------------|--|

#### 5.16.4.25 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     | <a href="#">optixReorder()</a> 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 |                                                                                                                                                                               |

#### 5.16.4.26 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.              |

#### 5.16.4.27 OptixExceptionFlags

enum [OptixExceptionFlags](#)

Exception flags.

See also [OptixPipelineCompileOptions::exceptionFlags](#), [OptixExceptionCodes](#)

Enumerator

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

## Enumerator

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

## 5.16.4.28 OptixGeometryFlags

enum [OptixGeometryFlags](#)

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

## Enumerator

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

## 5.16.4.29 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

|                                                 |                                         |
|-------------------------------------------------|-----------------------------------------|
| <code>OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE</code> | Ray hit the triangle on the front face. |
| <code>OPTIX_HIT_KIND_TRIANGLE_BACK_FACE</code>  | Ray hit the triangle on the back face.  |

#### 5.16.4.30 OptixIndicesFormat

enum `OptixIndicesFormat`

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

Enumerator

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

#### 5.16.4.31 OptixInstanceFlags

enum `OptixInstanceFlags`

Flags set on the `OptixInstance::flags`.

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

Enumerator

|                                                                |                                                                                                                                                                                                                      |
|----------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>OPTIX_INSTANCE_FLAG_NONE</code>                          | No special flag set.                                                                                                                                                                                                 |
| <code>OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING</code> | Prevent triangles from getting culled due to their orientation. Effectively ignores ray flags <code>OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES</code> and <code>OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES</code> .   |
| <code>OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING</code>          | Flip triangle orientation. This affects front/backface culling as well as the reported face in case of a hit.                                                                                                        |
| <code>OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT</code>                | Disable anyhit programs for all geometries of the instance. Can be overridden by <code>OPTIX_RAY_FLAG_ENFORCE_ANYHIT</code> . This flag is mutually exclusive with <code>OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT</code> . |

## Enumerator

|                                                    |                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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. Triangle GAS must be built with ALLOW_DISABLE_OPACITY_MICROMAPS for this to be valid. Clusters in a GAS must be build with OPTIX_CLUSTER_ACCEL_CLUSTER_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS for this to be valid. This flag overrides FORCE_OPACTIY_MIXROMAP_2_STATE instance and ray flags. |

## 5.16.4.32 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.   |

## 5.16.4.33 OptixMotionFlags

enum [OptixMotionFlags](#)

Enum to specify motion flags.

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

## Enumerator

|                                |
|--------------------------------|
| OPTIX_MOTION_FLAG_NONE         |
| OPTIX_MOTION_FLAG_START_VANISH |

Enumerator

|                              |
|------------------------------|
| OPTIX_MOTION_FLAG_END_VANISH |
|------------------------------|

#### 5.16.4.34 OptixOpacityMicromapArrayIndexingMode

enum [OptixOpacityMicromapArrayIndexingMode](#)

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

Enumerator

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

#### 5.16.4.35 OptixOpacityMicromapFlags

enum [OptixOpacityMicromapFlags](#)

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

Enumerator

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

#### 5.16.4.36 OptixOpacityMicromapFormat

enum [OptixOpacityMicromapFormat](#)

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

Enumerator

|                                    |                |
|------------------------------------|----------------|
| OPTIX_OPACITY_MICROMAP_FORMAT_NONE | invalid format |
|------------------------------------|----------------|

## Enumerator

|                                       |                                                                      |
|---------------------------------------|----------------------------------------------------------------------|
| OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE | 0: Transparent, 1: Opaque                                            |
| OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE | 0: Transparent, 1: Opaque, 2: Unknown-Transparent, 3: Unknown-Opaque |

## 5.16.4.37 OptixPayloadSemantics

enum `OptixPayloadSemantics`

Semantic flags for a single payload word.

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

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

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

## Enumerator

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

#### 5.16.4.38 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    |  |

#### 5.16.4.39 OptixPipelineSymbolMemcpyKind

enum [OptixPipelineSymbolMemcpyKind](#)

Flags used to interpret the source and target of memory copies when using [optixPipelineSymbolMemcpyAsync\(\)](#)

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

Enumerator

|                                               |
|-----------------------------------------------|
| OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_FROM_DEVICE |
| OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_FROM_HOST   |
| OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_TO_DEVICE   |
| OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_TO_HOST     |

#### 5.16.4.40 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    |

## Enumerator

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

## 5.16.4.41 OptixPrimitiveType

enum [OptixPrimitiveType](#)

Builtin primitive types.

## Enumerator

|                                                     |                                                                                    |
|-----------------------------------------------------|------------------------------------------------------------------------------------|
| OPTIX_PRIMITIVE_TYPE_CUSTOM                         | Custom primitive.                                                                  |
| OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE        | B-spline curve of degree 2 with circular cross-section.                            |
| OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE            | B-spline curve of degree 3 with circular cross-section.                            |
| OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR                   | Piecewise linear curve with circular cross-section.                                |
| OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM               | CatmullRom curve with circular cross-section.                                      |
| OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE         | B-spline curve of degree 2 with oriented, flat cross-section.                      |
| OPTIX_PRIMITIVE_TYPE_SPHERE                         | Sphere.                                                                            |
| OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER             | Bezier curve of degree 3 with circular cross-section.                              |
| OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE_ROCAPS | B-spline curve of degree 2 with circular cross-section, using rocaps intersection. |
| OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE_ROCAPS     | B-spline curve of degree 3 with circular cross-section, using rocaps intersection. |
| OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM_ROCAPS        | CatmullRom curve with circular cross-section, using rocaps intersection.           |
| OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER_ROCAPS      | Bezier curve of degree 3 with circular cross-section, using rocaps intersection.   |
| OPTIX_PRIMITIVE_TYPE_TRIANGLE                       | Triangle.                                                                          |

## 5.16.4.42 OptixPrimitiveTypeFlags

enum [OptixPrimitiveTypeFlags](#)

Builtin flags may be bitwise combined.

See also [OptixPipelineCompileOptions::usesPrimitiveTypeFlags](#)

## Enumerator

|                                                           |                                                                                    |
|-----------------------------------------------------------|------------------------------------------------------------------------------------|
| OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM                         | Custom primitive.                                                                  |
| OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE        | B-spline curve of degree 2 with circular cross-section.                            |
| OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE            | B-spline curve of degree 3 with circular cross-section.                            |
| OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR                   | Piecewise linear curve with circular cross-section.                                |
| OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM               | CatmullRom curve with circular cross-section.                                      |
| OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE         | B-spline curve of degree 2 with oriented, flat cross-section.                      |
| OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE                         | Sphere.                                                                            |
| OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER             | Bezier curve of degree 3 with circular cross-section.                              |
| OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE_ROCAPS | B-spline curve of degree 2 with circular cross-section, using rocaps intersection. |
| OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE_ROCAPS     | B-spline curve of degree 3 with circular cross-section, using rocaps intersection. |
| OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM_ROCAPS        | CatmullRom curve with circular cross-section, using rocaps intersection.           |
| OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER_ROCAPS      | Bezier curve of degree 3 with circular cross-section, using rocaps intersection.   |
| OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE                       | Triangle.                                                                          |

## 5.16.4.43 OptixProgramGroupFlags

enum [OptixProgramGroupFlags](#)

Flags for program groups.

## Enumerator

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

## 5.16.4.44 OptixProgramGroupKind

enum [OptixProgramGroupKind](#)

Distinguishes different kinds of program groups.

## Enumerator

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

## Enumerator

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

## 5.16.4.45 OptixQueryFunctionTableOptions

enum [OptixQueryFunctionTableOptions](#)

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

## Enumerator

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

## 5.16.4.46 OptixRayFlags

enum [OptixRayFlags](#)

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

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

## Enumerator

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

## Enumerator

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

## 5.16.4.47 OptixResult

enum `OptixResult`

Result codes returned from API functions.

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

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

#### Enumerator

|                                             |
|---------------------------------------------|
| OPTIX_SUCCESS                               |
| OPTIX_ERROR_INVALID_VALUE                   |
| OPTIX_ERROR_HOST_OUT_OF_MEMORY              |
| OPTIX_ERROR_INVALID_OPERATION               |
| OPTIX_ERROR_FILE_IO_ERROR                   |
| OPTIX_ERROR_INVALID_FILE_FORMAT             |
| OPTIX_ERROR_DISK_CACHE_INVALID_PATH         |
| OPTIX_ERROR_DISK_CACHE_PERMISSION_ERROR     |
| OPTIX_ERROR_DISK_CACHE_DATABASE_ERROR       |
| OPTIX_ERROR_DISK_CACHE_INVALID_DATA         |
| OPTIX_ERROR_LAUNCH_FAILURE                  |
| OPTIX_ERROR_INVALID_DEVICE_CONTEXT          |
| OPTIX_ERROR_CUDA_NOT_INITIALIZED            |
| OPTIX_ERROR_VALIDATION_FAILURE              |
| OPTIX_ERROR_INVALID_INPUT                   |
| OPTIX_ERROR_INVALID_LAUNCH_PARAMETER        |
| OPTIX_ERROR_INVALID_PAYLOAD_ACCESS          |
| OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS        |
| OPTIX_ERROR_INVALID_FUNCTION_USE            |
| OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS      |
| OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY |
| OPTIX_ERROR_PIPELINE_LINK_ERROR             |
| OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE     |
| OPTIX_ERROR_CREATION_CANCELED               |
| 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         |

## Enumerator

|                                            |
|--------------------------------------------|
| 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_INVALID_POINTER                |
| OPTIX_ERROR_SYMBOL_NOT_FOUND               |
| OPTIX_ERROR_CUDA_ERROR                     |
| OPTIX_ERROR_INTERNAL_ERROR                 |
| OPTIX_ERROR_UNKNOWN                        |

## 5.16.4.48 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                   |

## 5.16.4.49 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>              |

## 5.16.4.50 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. |

#### 5.16.4.51 OptixTraversableType

enum [OptixTraversableType](#)

Traversable Handles.

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

Enumerator

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

#### 5.16.4.52 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.   |

## Enumerator

|                               |  |
|-------------------------------|--|
| 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)
- template<typename HitState >  
static `__forceinline__ __device__ void optixGetWorldToObjectTransformMatrix` (const HitState &hs, float4 &m0, float4 &m1, float4 &m2)
- template<typename HitState >  
static `__forceinline__ __device__ void optixGetObjectToWorldTransformMatrix` (const HitState &hs, 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()

```

template<typename HitState >
static __forceinline__ __device__ void optix_impl
::optixGetObjectToWorldTransformMatrix (
 const HitState & hs,
 float4 & m0,
 float4 & m1,
 float4 & m2) [static]

```

#### 6.1.1.7 optixGetWorldToObjectTransformMatrix()

```

template<typename HitState >
static __forceinline__ __device__ void optix_impl
::optixGetWorldToObjectTransformMatrix (
 const HitState & hs,
 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 paramter to optixAccelBuild when doing a build (OPTIX\_BUILD\_OPERATION\_BUILD).

#### 7.2.2.3 tempUpdateSizeInBytes

size\_t OptixAccelBufferSizes::tempUpdateSizeInBytes

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

## 7.3 OptixAccelBuildOptions Struct Reference

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

### Public Attributes

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

### 7.3.1 Detailed Description

Build options for acceleration structures.

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

### 7.3.2 Member Data Documentation

#### 7.3.2.1 buildFlags

unsigned int [OptixAccelBuildOptions::buildFlags](#)

Combinations of [OptixBuildFlags](#).

#### 7.3.2.2 motionOptions

[OptixMotionOptions](#) [OptixAccelBuildOptions::motionOptions](#)

Options for motion.

#### 7.3.2.3 operation

[OptixBuildOperation](#) [OptixAccelBuildOptions::operation](#)

If OPTIX\_BUILD\_OPERATION\_UPDATE the output buffer is assumed to contain the result of a full build with OPTIX\_BUILD\_FLAG\_ALLOW\_UPDATE set and using the same number of primitives. It is updated incrementally to reflect the current position of the primitives. If a BLAS has been built with OPTIX\_BUILD\_FLAG\_ALLOW\_OPACITY\_MICROMAP\_UPDATE, new opacity micromap arrays and opacity micromap indices may be provided to the refit.

## 7.4 OptixAccelEmitDesc Struct Reference

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

### Public Attributes

- [CUdeviceptr](#) result
- [OptixAccelPropertyType](#) type

### 7.4.1 Detailed Description

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

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

## 7.4.2 Member Data Documentation

### 7.4.2.1 result

`CUdeviceptr OptixAccelEmitDesc::result`

Output buffer for the properties.

### 7.4.2.2 type

`OptixAccelPropertyType OptixAccelEmitDesc::type`

Requested property.

## 7.5 OptixBuildInput Struct Reference

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

### Public Attributes

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

### 7.5.1 Detailed Description

Build inputs.

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

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

## 7.5.2 Member Data Documentation

### 7.5.2.1

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

### 7.5.2.2 curveArray

`OptixBuildInputCurveArray OptixBuildInput::curveArray`

Curve inputs.

### 7.5.2.3 customPrimitiveArray

`OptixBuildInputCustomPrimitiveArray OptixBuildInput::customPrimitiveArray`

Custom primitive inputs.

### 7.5.2.4 instanceArray

`OptixBuildInputInstanceArray OptixBuildInput::instanceArray`

Instance and instance pointer inputs.

#### 7.5.2.5 pad

`char OptixBuildInput::pad[1024]`

#### 7.5.2.6 sphereArray

`OptixBuildInputSphereArray OptixBuildInput::sphereArray`

Sphere inputs.

#### 7.5.2.7 triangleArray

`OptixBuildInputTriangleArray OptixBuildInput::triangleArray`

Triangle inputs.

#### 7.5.2.8 type

`OptixBuildInputType OptixBuildInput::type`

The type of the build input.

## 7.6 OptixBuildInputCurveArray Struct Reference

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

### Public Attributes

- `OptixPrimitiveType curveType`
- unsigned int `numPrimitives`
- const `CUdeviceptr * vertexBuffers`
- unsigned int `numVertices`
- unsigned int `vertexStrideInBytes`
- const `CUdeviceptr * widthBuffers`
- unsigned int `widthStrideInBytes`
- const `CUdeviceptr * normalBuffers`
- unsigned int `normalStrideInBytes`
- `CUdeviceptr indexBuffer`
- unsigned int `indexStrideInBytes`
- unsigned int `flag`
- unsigned int `primitiveIndexOffset`
- unsigned int `endcapFlags`

### 7.6.1 Detailed Description

Curve inputs.

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

OptiX describes the curve array as a list of curve segments. The primitive id is the segment number. It is the user's responsibility to maintain a mapping between curves and curve segments. Each index buffer entry  $i = \text{indexBuffer}[\text{primid}]$  specifies the start of a curve segment, represented by  $d+1$

consecutive vertices in the vertex buffer, and d+1 consecutive widths in the width buffer. Width is interpolated the same way vertices are interpolated, that is, using the curve basis.

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

See also [OptixBuildInput::curveArray](#)

## 7.6.2 Member Data Documentation

### 7.6.2.1 curveType

[OptixPrimitiveType](#) OptixBuildInputCurveArray::curveType

Curve degree and basis.

See also [OptixPrimitiveType](#)

### 7.6.2.2 endcapFlags

unsigned int OptixBuildInputCurveArray::endcapFlags

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

### 7.6.2.3 flag

unsigned int OptixBuildInputCurveArray::flag

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

### 7.6.2.4 indexBuffer

[CUdeviceptr](#) OptixBuildInputCurveArray::indexBuffer

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

### 7.6.2.5 indexStrideInBytes

unsigned int OptixBuildInputCurveArray::indexStrideInBytes

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

### 7.6.2.6 normalBuffers

const [CUdeviceptr\\*](#) OptixBuildInputCurveArray::normalBuffers

Reserved for future use.

### 7.6.2.7 normalStrideInBytes

unsigned int OptixBuildInputCurveArray::normalStrideInBytes

Reserved for future use.

### 7.6.2.8 numPrimitives

unsigned int OptixBuildInputCurveArray::numPrimitives

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

#### 7.6.2.9 numVertices

```
unsigned int OptixBuildInputCurveArray::numVertices
```

Number of vertices in each buffer in vertexBuffers.

#### 7.6.2.10 primitiveIndexOffset

```
unsigned int OptixBuildInputCurveArray::primitiveIndexOffset
```

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

#### 7.6.2.11 vertexBuffers

```
const CUdeviceptr* OptixBuildInputCurveArray::vertexBuffers
```

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

#### 7.6.2.12 vertexStrideInBytes

```
unsigned int OptixBuildInputCurveArray::vertexStrideInBytes
```

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

#### 7.6.2.13 widthBuffers

```
const CUdeviceptr* OptixBuildInputCurveArray::widthBuffers
```

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

#### 7.6.2.14 widthStrideInBytes

```
unsigned int OptixBuildInputCurveArray::widthStrideInBytes
```

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

### 7.7 OptixBuildInputCustomPrimitiveArray Struct Reference

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

#### Public Attributes

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

## 7.7.1 Detailed Description

Custom primitive inputs.

See also [OptixBuildInput::customPrimitiveArray](#)

## 7.7.2 Member Data Documentation

### 7.7.2.1 aabbBuffers

`const CUdeviceptr* OptixBuildInputCustomPrimitiveArray::aabbBuffers`

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

### 7.7.2.2 flags

`const unsigned int* OptixBuildInputCustomPrimitiveArray::flags`

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

### 7.7.2.3 numPrimitives

`unsigned int OptixBuildInputCustomPrimitiveArray::numPrimitives`

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

### 7.7.2.4 numSbtRecords

`unsigned int OptixBuildInputCustomPrimitiveArray::numSbtRecords`

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

### 7.7.2.5 primitiveIndexOffset

`unsigned int OptixBuildInputCustomPrimitiveArray::primitiveIndexOffset`

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

### 7.7.2.6 sbtIndexOffsetBuffer

`CUdeviceptr OptixBuildInputCustomPrimitiveArray::sbtIndexOffsetBuffer`

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

### 7.7.2.7 sbtIndexOffsetSizeInBytes

`unsigned int OptixBuildInputCustomPrimitiveArray::sbtIndexOffsetSizeInBytes`

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

### 7.7.2.8 sbtIndexOffsetStrideInBytes

`unsigned int OptixBuildInputCustomPrimitiveArray`

`::sbtIndexOffsetStrideInBytes`

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

### 7.7.2.9 strideInBytes

`unsigned int OptixBuildInputCustomPrimitiveArray::strideInBytes`

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

## 7.8 OptixBuildInputInstanceArray Struct Reference

`#include <optix_types.h>`

### Public Attributes

- [CUdeviceptr](#) `instances`
- unsigned int `numInstances`
- unsigned int `instanceStride`

### 7.8.1 Detailed Description

Instance and instance pointer inputs.

See also [OptixBuildInput::instanceArray](#)

### 7.8.2 Member Data Documentation

#### 7.8.2.1 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`.

#### 7.8.2.2 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)`

#### 7.8.2.3 numInstances

`unsigned int OptixBuildInputInstanceArray::numInstances`

Number of elements in [OptixBuildInputInstanceArray::instances](#).

## 7.9 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

## 7.9.1 Member Data Documentation

### 7.9.1.1 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](#).

### 7.9.1.2 indexingMode

[OptixOpacityMicromapArrayIndexingMode](#) [OptixBuildInputOpacityMicromap::indexingMode](#)

Indexing mode of triangle to opacity micromap array mapping.

### 7.9.1.3 indexOffset

unsigned int [OptixBuildInputOpacityMicromap::indexOffset](#)

Constant offset to non-negative opacity micromap indices.

### 7.9.1.4 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](#).

### 7.9.1.5 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](#).

### 7.9.1.6 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.

### 7.9.1.7 numMicromapUsageCounts

`unsigned int OptixBuildInputOpacityMicromap::numMicromapUsageCounts`

Number of [OptixOpacityMicromapUsageCount](#).

### 7.9.1.8 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`.

## 7.10 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`

### 7.10.1 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](#)

## 7.10.2 Member Data Documentation

### 7.10.2.1 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`.

### 7.10.2.2 numSbtRecords

`unsigned int OptixBuildInputSphereArray::numSbtRecords`

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

### 7.10.2.3 numVertices

`unsigned int OptixBuildInputSphereArray::numVertices`

Number of vertices in each buffer in `vertexBuffers`.

### 7.10.2.4 primitiveIndexOffset

`unsigned int OptixBuildInputSphereArray::primitiveIndexOffset`

Primitive index bias, applied in `optixGetPrimitiveIndex()`. Sum of `primitiveIndexOffset` and number of primitives must not overflow 32bits.

### 7.10.2.5 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.

### 7.10.2.6 radiusStrideInBytes

`unsigned int OptixBuildInputSphereArray::radiusStrideInBytes`

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

### 7.10.2.7 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.

### 7.10.2.8 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).

### 7.10.2.9 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`).

### 7.10.2.10 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`.

### 7.10.2.11 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).

### 7.10.2.12 vertexStrideInBytes

`unsigned int OptixBuildInputSphereArray::vertexStrideInBytes`

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

## 7.11 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`

### 7.11.1 Detailed Description

Triangle inputs.

See also `OptixBuildInput::triangleArray`

### 7.11.2 Member Data Documentation

#### 7.11.2.1 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`.

### 7.11.2.2 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.

### 7.11.2.3 indexFormat

`OptixIndicesFormat` `OptixBuildInputTriangleArray::indexFormat`

See also `OptixIndicesFormat`

### 7.11.2.4 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`.

### 7.11.2.5 numIndexTriplets

`unsigned int` `OptixBuildInputTriangleArray::numIndexTriplets`

Size of array in `OptixBuildInputTriangleArray::indexBuffer`. For build, needs to be zero if `indexBuffer` is `nullptr`.

### 7.11.2.6 numSbtRecords

`unsigned int` `OptixBuildInputTriangleArray::numSbtRecords`

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

### 7.11.2.7 numVertices

`unsigned int` `OptixBuildInputTriangleArray::numVertices`

Number of vertices in each of buffer in `OptixBuildInputTriangleArray::vertexBuffers`.

### 7.11.2.8 opacityMicromap

`OptixBuildInputOpacityMicromap` `OptixBuildInputTriangleArray::opacityMicromap`

Optional opacity micromap inputs.

### 7.11.2.9 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`.

### 7.11.2.10 primitiveIndexOffset

`unsigned int OptixBuildInputTriangleArray::primitiveIndexOffset`

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

### 7.11.2.11 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.

### 7.11.2.12 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).

### 7.11.2.13 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`).

### 7.11.2.14 transformFormat

`OptixTransformFormat OptixBuildInputTriangleArray::transformFormat`

See also [OptixTransformFormat](#)

### 7.11.2.15 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.

### 7.11.2.16 vertexFormat

`OptixVertexFormat OptixBuildInputTriangleArray::vertexFormat`

See also [OptixVertexFormat](#)

### 7.11.2.17 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`.

## 7.12 OptixBuiltinISOOptions Struct Reference

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

### Public Attributes

- [OptixPrimitiveType](#) `builtinISModuleType`
- `int` `usesMotionBlur`
- `unsigned int` `buildFlags`
- `unsigned int` `curveEndcapFlags`

### 7.12.1 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\(\)](#)

### 7.12.2 Member Data Documentation

#### 7.12.2.1 buildFlags

```
unsigned int OptixBuiltinISOOptions::buildFlags
```

Build flags, see [OptixBuildFlags](#).

#### 7.12.2.2 builtinISModuleType

```
OptixPrimitiveType OptixBuiltinISOOptions::builtinISModuleType
```

#### 7.12.2.3 curveEndcapFlags

```
unsigned int OptixBuiltinISOOptions::curveEndcapFlags
```

End cap properties of curves, see [OptixCurveEndcapFlags](#), 0 for non-curve types.

#### 7.12.2.4 usesMotionBlur

```
int OptixBuiltinISOOptions::usesMotionBlur
```

Boolean value indicating whether vertex motion blur is used (but not motion transform blur).

## 7.13 OptixClusterAccelBuildInput Struct Reference

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

### Public Attributes

- [OptixClusterAccelBuildType](#) `type`
- `union {`
  - [OptixClusterAccelBuildInputTriangles](#) `triangles`
  - [OptixClusterAccelBuildInputClusters](#) `clusters`
  - [OptixClusterAccelBuildInputGrids](#) `grids`
- `};`

### 7.13.1 Member Data Documentation

#### 7.13.1.1

`union { ... } OptixClusterAccelBuildInput::@5`

#### 7.13.1.2 clusters

[OptixClusterAccelBuildInputClusters](#) `OptixClusterAccelBuildInput::clusters`

Used for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_GASES\_FROM\_CLUSTERS type builds.

#### 7.13.1.3 grids

[OptixClusterAccelBuildInputGrids](#) `OptixClusterAccelBuildInput::grids`

Used for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_TEMPLATES\_FROM\_GRIDS type builds.

#### 7.13.1.4 triangles

[OptixClusterAccelBuildInputTriangles](#) `OptixClusterAccelBuildInput::triangles`

Used for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_CLUSTERS\_FROM\_TRIANGLES, OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_TEMPLATES\_FROM\_TRIANGLES, OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_CLUSTERS\_FROM\_TEMPLATES type builds.

#### 7.13.1.5 type

[OptixClusterAccelBuildType](#) `OptixClusterAccelBuildInput::type`

## 7.14 OptixClusterAccelBuildInputClusters Struct Reference

`#include <optix_types.h>`

### Public Attributes

- [OptixClusterAccelBuildFlags](#) `flags`
- unsigned int `maxArgCount`
- unsigned int `maxTotalClusterCount`
- unsigned int `maxClusterCountPerArg`

### 7.14.1 Member Data Documentation

#### 7.14.1.1 flags

[OptixClusterAccelBuildFlags](#) `OptixClusterAccelBuildInputClusters::flags`

#### 7.14.1.2 maxArgCount

unsigned int `OptixClusterAccelBuildInputClusters::maxArgCount`

Max number of [OptixClusterAccelBuildInputClustersArgs](#) provided at build time for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_GASES\_FROM\_CLUSTERS.

#### 7.14.1.3 maxClusterCountPerArg

unsigned int `OptixClusterAccelBuildInputClusters::maxClusterCountPerArg`

#### 7.14.1.4 maxTotalClusterCount

unsigned int OptixClusterAccelBuildInputClusters::maxTotalClusterCount

### 7.15 OptixClusterAccelBuildInputClustersArgs Struct Reference

#include <optix\_types.h>

#### Public Attributes

- unsigned int [clusterHandlesCount](#)
- unsigned int [clusterHandlesBufferStrideInBytes](#)
- [CUdeviceptr](#) [clusterHandlesBuffer](#)

#### 7.15.1 Detailed Description

Device data, args provided for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_GASES\_FROM\_CLUSTERS builds.

#### 7.15.2 Member Data Documentation

##### 7.15.2.1 clusterHandlesBuffer

[CUdeviceptr](#) OptixClusterAccelBuildInputClustersArgs::clusterHandlesBuffer

The clusterHandlesBuffer can come directly from CLAS builds output via [OptixClusterAccelBuildModeDescImplicitDest::outputHandlesBuffer](#) or [OptixClusterAccelBuildModeDescExplicitDest::outputHandlesBuffer](#).

##### 7.15.2.2 clusterHandlesBufferStrideInBytes

unsigned int OptixClusterAccelBuildInputClustersArgs::clusterHandlesBufferStrideInBytes

##### 7.15.2.3 clusterHandlesCount

unsigned int OptixClusterAccelBuildInputClustersArgs::clusterHandlesCount

Number of CLAS input to the BLAS build (size of the clusterHandles buffer)

### 7.16 OptixClusterAccelBuildInputGrids Struct Reference

#include <optix\_types.h>

#### Public Attributes

- [OptixClusterAccelBuildFlags](#) [flags](#)
- unsigned int [maxArgCount](#)
- [OptixVertexFormat](#) [vertexFormat](#)
- unsigned int [maxSbtIndexValue](#)
- unsigned int [maxWidth](#)
- unsigned int [maxHeight](#)

#### 7.16.1 Member Data Documentation

##### 7.16.1.1 flags

[OptixClusterAccelBuildFlags](#) OptixClusterAccelBuildInputGrids::flags

### 7.16.1.2 maxArgCount

`unsigned int OptixClusterAccelBuildInputGrids::maxArgCount`

### 7.16.1.3 maxHeight

`unsigned int OptixClusterAccelBuildInputGrids::maxHeight`

The maximum number of edge segments along the height of any grid.

### 7.16.1.4 maxSbtIndexValue

`unsigned int OptixClusterAccelBuildInputGrids::maxSbtIndexValue`

The maximum used SBT index over all clusters. This must include the base SBT offset (`::basePrimitiveInfo`), any potential per primitive offset (`::primitiveInfoBuffer`), as well as a potential offset at template instantiation (`OptixClusterAccelBuildInputTemplatesArgs::sbtIndexOffset`)

### 7.16.1.5 maxWidth

`unsigned int OptixClusterAccelBuildInputGrids::maxWidth`

The maximum number of edge segments along the width of any grid.

### 7.16.1.6 vertexFormat

`OptixVertexFormat OptixClusterAccelBuildInputGrids::vertexFormat`

`OptixVertexFormat` (see documentation for supported formats)

## 7.17 OptixClusterAccelBuildInputGridsArgs Struct Reference

`#include <optix_types.h>`

### Public Attributes

- `unsigned int baseClusterId`
- `unsigned int clusterFlags`
- `OptixClusterAccelPrimitiveInfo basePrimitiveInfo`
- `unsigned int positionTruncateBitCount: 6`
- `unsigned int reserved: 26`
- `unsigned char dimensions [2]`
- `unsigned short reserved2`

### 7.17.1 Detailed Description

Device data, args provided for `OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS` builds.

### 7.17.2 Member Data Documentation

#### 7.17.2.1 baseClusterId

`unsigned int OptixClusterAccelBuildInputGridsArgs::baseClusterId`

32-bit user-defined ID, serves as a base value for the template and can be offset at template instantiation (see `OptixClusterAccelBuildInputTemplatesArgs::clusterIdOffset`)

### 7.17.2.2 basePrimitiveInfo

[OptixClusterAccelPrimitiveInfo](#) [OptixClusterAccelBuildInputGridsArgs](#)  
::basePrimitiveInfo

Applied to all triangles in cluster.

### 7.17.2.3 clusterFlags

unsigned int [OptixClusterAccelBuildInputGridsArgs](#)::clusterFlags

Combination of [OptixClusterAccelClusterFlags](#).

### 7.17.2.4 dimensions

unsigned char [OptixClusterAccelBuildInputGridsArgs](#)::dimensions[2]

Resolution of the 2D grid, max value per dimension can be queried.

### 7.17.2.5 positionTruncateBitCount

unsigned int [OptixClusterAccelBuildInputGridsArgs](#)::positionTruncateBitCount

See [OptixClusterAccelBuildInputTrianglesArgs::positionTruncateBitCount](#).

### 7.17.2.6 reserved

unsigned int [OptixClusterAccelBuildInputGridsArgs](#)::reserved

### 7.17.2.7 reserved2

unsigned short [OptixClusterAccelBuildInputGridsArgs](#)::reserved2

## 7.18 OptixClusterAccelBuildInputTemplatesArgs Struct Reference

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

### Public Attributes

- unsigned int [clusterIdOffset](#)
- unsigned int [sbtIndexOffset](#)
- CUdeviceptr [clusterTemplate](#)
- CUdeviceptr [vertexBuffer](#)
- unsigned int [vertexStrideInBytes](#)
- unsigned int [reserved](#)

### 7.18.1 Detailed Description

Device data, args provided for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_CLUSTERS\_FROM\_TEMPLATES builds.

### 7.18.2 Member Data Documentation

#### 7.18.2.1 clusterIdOffset

unsigned int [OptixClusterAccelBuildInputTemplatesArgs](#)::clusterIdOffset

Offset applied to template baseClusterId, effective clusterId = clusterTemplate.baseClusterId + clusterIdOffset. Either may be 0.

### 7.18.2.2 clusterTemplate

`CUdeviceptr` `OptixClusterAccelBuildInputTemplatesArgs::clusterTemplate`

Opaque pointer to the template.

### 7.18.2.3 reserved

`unsigned int` `OptixClusterAccelBuildInputTemplatesArgs::reserved`

### 7.18.2.4 sbtIndexOffset

`unsigned int` `OptixClusterAccelBuildInputTemplatesArgs::sbtIndexOffset`

Offset to base sbtIndex from template creation (which may define a constant or per-triangle base sbtIndex), final sbt index is also limited to fit into 24b.

### 7.18.2.5 vertexBuffer

`CUdeviceptr` `OptixClusterAccelBuildInputTemplatesArgs::vertexBuffer`

The vertex data to use to instantiate the template; vertex order must match that of template creation. For templates created from grids, see documentation.

### 7.18.2.6 vertexStrideInBytes

`unsigned int` `OptixClusterAccelBuildInputTemplatesArgs::vertexStrideInBytes`

Stride between elements in vertex buffer. Stride 0 -> natural stride.

## 7.19 OptixClusterAccelBuildInputTriangles Struct Reference

`#include <optix_types.h>`

### Public Attributes

- `OptixClusterAccelBuildFlags` `flags`
- `unsigned int` `maxArgCount`
- `OptixVertexFormat` `vertexFormat`
- `unsigned int` `maxSbtIndexValue`
- `unsigned int` `maxUniqueSbtIndexCountPerArg`
- `unsigned int` `maxTriangleCountPerArg`
- `unsigned int` `maxVertexCountPerArg`
- `unsigned int` `maxTotalTriangleCount`
- `unsigned int` `maxTotalVertexCount`
- `unsigned int` `minPositionTruncateBitCount`

### 7.19.1 Member Data Documentation

#### 7.19.1.1 flags

`OptixClusterAccelBuildFlags` `OptixClusterAccelBuildInputTriangles::flags`

#### 7.19.1.2 maxArgCount

`unsigned int` `OptixClusterAccelBuildInputTriangles::maxArgCount`

Max number of `OptixClusterAccelBuildInputTrianglesArgs` provided at build time for `OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES` and `OPTIX_CLUSTER_ACCEL_`

BUILD\_TYPE\_TEMPLATES\_FROM\_TRIANGLES. Max number of [OptixClusterAccelBuildInputTemplatesArgs](#) provided at build time for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_CLUSTERS\_FROM\_TEMPLATES.

### 7.19.1.3 maxSbtIndexValue

`unsigned int OptixClusterAccelBuildInputTriangles::maxSbtIndexValue`

The maximum used sbt index over all clusters; This must include the base sbt offset (`::basePrimitiveInfo`), any potential per primitive offset (`::primitiveInfoBuffer`), as well as a potential offset at template instantiation ([OptixClusterAccelBuildInputTemplatesArgs::sbtIndexOffset](#))

### 7.19.1.4 maxTotalTriangleCount

`unsigned int OptixClusterAccelBuildInputTriangles::maxTotalTriangleCount`

Optional, upper bound on the number of triangles over all Args, `maxTriangleCountPerArg * maxArgCount` otherwise.

### 7.19.1.5 maxTotalVertexCount

`unsigned int OptixClusterAccelBuildInputTriangles::maxTotalVertexCount`

Optional, upper bound on the number of vertices over all Args, `maxVertexCountPerArg * maxArgCount` otherwise.

### 7.19.1.6 maxTriangleCountPerArg

`unsigned int OptixClusterAccelBuildInputTriangles::maxTriangleCountPerArg`

Upper bound on the number of triangles per Arg.

### 7.19.1.7 maxUniqueSbtIndexCountPerArg

`unsigned int OptixClusterAccelBuildInputTriangles::maxUniqueSbtIndexCountPerArg`

Number of unique SBT indices per cluster. If the cluster has the same SBT index for all its triangles, this value is 1.

### 7.19.1.8 maxVertexCountPerArg

`unsigned int OptixClusterAccelBuildInputTriangles::maxVertexCountPerArg`

Upper bound on the number of vertices per Arg.

### 7.19.1.9 minPositionTruncateBitCount

`unsigned int OptixClusterAccelBuildInputTriangles::minPositionTruncateBitCount`

Lower bound on the number of bits being truncated of the vertex positions.

### 7.19.1.10 vertexFormat

[OptixVertexFormat](#) `OptixClusterAccelBuildInputTriangles::vertexFormat`

[OptixVertexFormat](#) (see documentation for supported formats)

## 7.20 OptixClusterAccelBuildInputTrianglesArgs Struct Reference

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

### Public Attributes

- unsigned int `clusterId`
- unsigned int `clusterFlags`
- unsigned int `triangleCount`: 9
- unsigned int `vertexCount`: 9
- unsigned int `positionTruncateBitCount`: 6
- unsigned int `indexFormat`: 4
- unsigned int `opacityMicromapIndexFormat`: 4
- `OptixClusterAccelPrimitiveInfo` `basePrimitiveInfo`
- unsigned short `indexBufferStrideInBytes`
- unsigned short `vertexBufferStrideInBytes`
- unsigned short `primitiveInfoBufferStrideInBytes`
- unsigned short `opacityMicromapIndexBufferStrideInBytes`
- `CUdeviceptr` `indexBuffer`
- `CUdeviceptr` `vertexBuffer`
- `CUdeviceptr` `primitiveInfoBuffer`
- `CUdeviceptr` `opacityMicromapArray`
- `CUdeviceptr` `opacityMicromapIndexBuffer`
- `CUdeviceptr` `instantiationBoundingBoxLimit`

### 7.20.1 Detailed Description

Device data, args provided for `OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES` builds and `OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES` builds.

### 7.20.2 Member Data Documentation

#### 7.20.2.1 `basePrimitiveInfo`

`OptixClusterAccelPrimitiveInfo` `OptixClusterAccelBuildInputTrianglesArgs::basePrimitiveInfo`

Applied to all triangles in cluster. Additional per triangle flags can be specified in `PrimitiveInfoBuffer`.

#### 7.20.2.2 `clusterFlags`

unsigned int `OptixClusterAccelBuildInputTrianglesArgs::clusterFlags`

Combination of `OptixClusterAccelClusterFlags`.

#### 7.20.2.3 `clusterId`

unsigned int `OptixClusterAccelBuildInputTrianglesArgs::clusterId`

32-bit user-defined ID, for template creation acts as the `baseClusterId` and can be offset at template instantiation (see `OptixClusterAccelBuildInputTemplatesArgs::clusterIdOffset`)

#### 7.20.2.4 `indexBuffer`

`CUdeviceptr` `OptixClusterAccelBuildInputTrianglesArgs::indexBuffer`

Triplets of vertex indices into `vertexBuffer` per triangle. Must contain  $3 * \text{triangleCount}$  indices.

### 7.20.2.5 indexBufferStrideInBytes

unsigned short OptixClusterAccelBuildInputTrianglesArgs  
::indexBufferStrideInBytes

Stride between elements in index buffer. Stride 0 -> natural stride.

### 7.20.2.6 indexFormat

unsigned int OptixClusterAccelBuildInputTrianglesArgs::indexFormat

Can use OptixClusterAccelIndicesFormat as helper to set value: 1, 2, or 4 bytes-wide indices.

### 7.20.2.7 instantiationBoundingBoxLimit

CUdeviceptr OptixClusterAccelBuildInputTrianglesArgs  
::instantiationBoundingBoxLimit

Optional with OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_TEMPLATES\_FROM\_TRIANGLES, 32-byte-aligned pointer to [OptixAabb](#), one per cluster, limiting the extent of each cluster. Vertices provided for template instantiation must not be outside the bounding box. Providing a bounding box may improve compression (reduced CLAS size) as well as trace performance. Ignored for OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_CLUSTERS\_FROM\_TRIANGLES.

### 7.20.2.8 opacityMicromapArray

CUdeviceptr OptixClusterAccelBuildInputTrianglesArgs::opacityMicromapArray

Optional, needs to be set if OMMs are used.

### 7.20.2.9 opacityMicromapIndexBuffer

CUdeviceptr OptixClusterAccelBuildInputTrianglesArgs  
::opacityMicromapIndexBuffer

Optional, needs to be set if OMMs are used.

### 7.20.2.10 opacityMicromapIndexBufferStrideInBytes

unsigned short OptixClusterAccelBuildInputTrianglesArgs  
::opacityMicromapIndexBufferStrideInBytes

Stride between elements in omm index buffer. Stride 0 -> natural stride.

### 7.20.2.11 opacityMicromapIndexFormat

unsigned int OptixClusterAccelBuildInputTrianglesArgs  
::opacityMicromapIndexFormat

Can use OptixClusterAccelIndicesFormat as helper to set value: 1, 2, or 4 bytes-wide indices.

### 7.20.2.12 positionTruncateBitCount

unsigned int OptixClusterAccelBuildInputTrianglesArgs  
::positionTruncateBitCount

Number of LSB in mantissa that are dropped (0 means don't drop any) for float32 positions. Other formats are first converted to float32 before dropping bits. Builder will drop bits when building CLAS / instantiating cluster templates (no need to truncate the input before build).

### 7.20.2.13 primitiveInfoBuffer

`CUdeviceptr` `OptixClusterAccelBuildInputTrianglesArgs::primitiveInfoBuffer`

Optional, per primitive array of `OptixClusterAccelPrimitiveInfo`.

### 7.20.2.14 primitiveInfoBufferStrideInBytes

`unsigned short` `OptixClusterAccelBuildInputTrianglesArgs::primitiveInfoBufferStrideInBytes`

Stride between elements in primitive info buffer. Stride 0 -> natural stride.

### 7.20.2.15 triangleCount

`unsigned int` `OptixClusterAccelBuildInputTrianglesArgs::triangleCount`

Number of triangles for cluster / cluster template, max value can be queried.

### 7.20.2.16 vertexBuffer

`CUdeviceptr` `OptixClusterAccelBuildInputTrianglesArgs::vertexBuffer`

`vertexBuffer` is mandatory when using `OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES`. Optional with `OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES` and when specified provide example "hint" vertices for templates; actual vertices are specified at template instantiation. It is typically useful to provide vertices for template creation in scenarios such as animation, where the relative locality of vertices is expected to be similar between the template creation and instantiation.

### 7.20.2.17 vertexBufferStrideInBytes

`unsigned short` `OptixClusterAccelBuildInputTrianglesArgs::vertexBufferStrideInBytes`

Stride between elements in vertex buffer. Stride 0 -> natural stride.

### 7.20.2.18 vertexCount

`unsigned int` `OptixClusterAccelBuildInputTrianglesArgs::vertexCount`

Number of vertices shared by triangles for cluster / cluster template, max value can be queried.

## 7.21 OptixClusterAccelBuildModeDesc Struct Reference

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

### Public Attributes

- `OptixClusterAccelBuildMode` `mode`
- `union {`
  - `OptixClusterAccelBuildModeDescImplicitDest` `implicitDest`
  - `OptixClusterAccelBuildModeDescExplicitDest` `explicitDest`
  - `OptixClusterAccelBuildModeDescGetSize` `getSize`
- `};`

## 7.21.1 Member Data Documentation

### 7.21.1.1

```
union { ... } OptixClusterAccelBuildModeDesc::@7
```

### 7.21.1.2 explicitDest

```
OptixClusterAccelBuildModeDescExplicitDest OptixClusterAccelBuildModeDesc
::explicitDest
```

### 7.21.1.3 getSize

```
OptixClusterAccelBuildModeDescGetSize OptixClusterAccelBuildModeDesc
::getSize
```

### 7.21.1.4 implicitDest

```
OptixClusterAccelBuildModeDescImplicitDest OptixClusterAccelBuildModeDesc
::implicitDest
```

### 7.21.1.5 mode

```
OptixClusterAccelBuildMode OptixClusterAccelBuildModeDesc::mode
```

## 7.22 OptixClusterAccelBuildModeDescExplicitDest Struct Reference

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

### Public Attributes

- CUdeviceptr tempBuffer
- size\_t tempBufferSizeInBytes
- CUdeviceptr destAddressesBuffer
- unsigned int destAddressesStrideInBytes
- CUdeviceptr outputHandlesBuffer
- unsigned int outputHandlesStrideInBytes
- CUdeviceptr outputSizesBuffer
- unsigned int outputSizesStrideInBytes

## 7.22.1 Member Data Documentation

### 7.22.1.1 destAddressesBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescExplicitDest::destAddressesBuffer
```

Entries must be aligned according to the output type.

### 7.22.1.2 destAddressesStrideInBytes

```
unsigned int OptixClusterAccelBuildModeDescExplicitDest
::destAddressesStrideInBytes
```

Minimum 8, Stride of 0 implies natural stride of 8B.

### 7.22.1.3 outputHandlesBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescExplicitDest::outputHandlesBuffer
```

TraversableHandle for GAS, pointer for cluster and template outputs, can be the same as destAddresses in which case they will overwrite the input.

#### 7.22.1.4 outputHandlesStrideInBytes

unsigned int OptixClusterAccelBuildModeDescExplicitDest::outputHandlesStrideInBytes

Minimum 8, Stride of 0 implies natural stride of 8B.

#### 7.22.1.5 outputSizesBuffer

CUdeviceptr OptixClusterAccelBuildModeDescExplicitDest::outputSizesBuffer

Optional, uint32 array (4 byte aligned)

#### 7.22.1.6 outputSizesStrideInBytes

unsigned int OptixClusterAccelBuildModeDescExplicitDest::outputSizesStrideInBytes

Minimum 4, Stride of 0 implies natural stride of 4B.

#### 7.22.1.7 tempBuffer

CUdeviceptr OptixClusterAccelBuildModeDescExplicitDest::tempBuffer

128-byte aligned, see OPTIX\_ACCEL\_BUFFER\_BYTE\_ALIGNMENT

#### 7.22.1.8 tempBufferSizeInBytes

size\_t OptixClusterAccelBuildModeDescExplicitDest::tempBufferSizeInBytes

### 7.23 OptixClusterAccelBuildModeDescGetSize Struct Reference

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

#### Public Attributes

- CUdeviceptr outputSizesBuffer
- unsigned int outputSizesStrideInBytes
- CUdeviceptr tempBuffer
- size\_t tempBufferSizeInBytes

#### 7.23.1 Member Data Documentation

##### 7.23.1.1 outputSizesBuffer

CUdeviceptr OptixClusterAccelBuildModeDescGetSize::outputSizesBuffer

Mandatory, uint32 array (4 byte aligned)

##### 7.23.1.2 outputSizesStrideInBytes

unsigned int OptixClusterAccelBuildModeDescGetSize::outputSizesStrideInBytes

Minimum 4, Stride of 0 implies natural stride of 4B.

### 7.23.1.3 tempBuffer

`CUdeviceptr` OptixClusterAccelBuildModeDescGetSize::tempBuffer

128-byte aligned, see OPTIX\_ACCEL\_BUFFER\_BYTE\_ALIGNMENT

### 7.23.1.4 tempBufferSizeInBytes

`size_t` OptixClusterAccelBuildModeDescGetSize::tempBufferSizeInBytes

## 7.24 OptixClusterAccelBuildModeDescImplicitDest Struct Reference

`#include <optix_types.h>`

### Public Attributes

- `CUdeviceptr` outputBuffer
- `size_t` outputBufferSizeInBytes
- `CUdeviceptr` tempBuffer
- `size_t` tempBufferSizeInBytes
- `CUdeviceptr` outputHandlesBuffer
- `unsigned int` outputHandlesStrideInBytes
- `CUdeviceptr` outputSizesBuffer
- `unsigned int` outputSizesStrideInBytes

## 7.24.1 Member Data Documentation

### 7.24.1.1 outputBuffer

`CUdeviceptr` OptixClusterAccelBuildModeDescImplicitDest::outputBuffer

alignment of outputBuffer must match result type. Clusters: 128 bytes Templates: 32 bytes GASes: 128 bytes, see OPTIX\_ACCEL\_BUFFER\_BYTE\_ALIGNMENT

### 7.24.1.2 outputBufferSizeInBytes

`size_t` OptixClusterAccelBuildModeDescImplicitDest::outputBufferSizeInBytes

size of outputHandlesBuffer is outputHandlesStrideInBytes \* number of inputs specified with either argCount or maxArgCount

### 7.24.1.3 outputHandlesBuffer

`CUdeviceptr` OptixClusterAccelBuildModeDescImplicitDest::outputHandlesBuffer

TraversableHandle for GAS, pointer for cluster and template outputs.

### 7.24.1.4 outputHandlesStrideInBytes

`unsigned int` OptixClusterAccelBuildModeDescImplicitDest::outputHandlesStrideInBytes

Minimum 8, Stride of 0 implies natural stride of 8B.

### 7.24.1.5 outputSizesBuffer

`CUdeviceptr` OptixClusterAccelBuildModeDescImplicitDest::outputSizesBuffer

Optional, uint32 array (4 byte aligned)

### 7.24.1.6 outputSizesStrideInBytes

unsigned int OptixClusterAccelBuildModeDescImplicitDest::outputSizesStrideInBytes

Minimum 4, Stride of 0 implies natural stride of 4B.

### 7.24.1.7 tempBuffer

CUdeviceptr OptixClusterAccelBuildModeDescImplicitDest::tempBuffer

128-byte aligned, see OPTIX\_ACCEL\_BUFFER\_BYTE\_ALIGNMENT

### 7.24.1.8 tempBufferSizeInBytes

size\_t OptixClusterAccelBuildModeDescImplicitDest::tempBufferSizeInBytes

## 7.25 OptixClusterAccelPrimitiveInfo Struct Reference

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

### Public Attributes

- unsigned int [sbtIndex](#): 24
- unsigned int [reserved](#): 5
- unsigned int [primitiveFlags](#): 3

## 7.25.1 Member Data Documentation

### 7.25.1.1 primitiveFlags

unsigned int OptixClusterAccelPrimitiveInfo::primitiveFlags

Combination of OptixClusterAccelPrimitiveFlags.

### 7.25.1.2 reserved

unsigned int OptixClusterAccelPrimitiveInfo::reserved

### 7.25.1.3 sbtIndex

unsigned int OptixClusterAccelPrimitiveInfo::sbtIndex

## 7.26 OptixCoopVec< T, N > Class Template Reference

```
#include <optix_device.h>
```

### Public Types

- using [value\\_type](#) = T

### Public Member Functions

- [\\_\\_forceinline\\_\\_ \\_\\_device\\_\\_ OptixCoopVec \(\)](#)
- [\\_\\_forceinline\\_\\_ \\_\\_device\\_\\_ OptixCoopVec \(const value\\_type &val\)](#)
- [\\_\\_forceinline\\_\\_ \\_\\_device\\_\\_ const value\\_type & operator\[\] \(unsigned int index\) const](#)
- [\\_\\_forceinline\\_\\_ \\_\\_device\\_\\_ value\\_type & operator\[\] \(unsigned int index\)](#)
- [\\_\\_forceinline\\_\\_ \\_\\_device\\_\\_ const value\\_type \\* data \(\) const](#)
- [\\_\\_forceinline\\_\\_ \\_\\_device\\_\\_ value\\_type \\* data \(\)](#)

## Static Public Attributes

- static const unsigned int `size` = N

## Protected Attributes

- `value_type m_data` [`size`]

### 7.26.1 Detailed Description

**template<typename T, unsigned int N>**

**class OptixCoopVec< T, N >**

The API does not require the use of this class specifically, but it must define a certain interface as spelled out by the public members of the class. Note that not all types of T are supported. Only 8 and 32 bit signed and unsigned integral types along with 16 and 32 bit floating point values.

### 7.26.2 Member Typedef Documentation

#### 7.26.2.1 value\_type

```
template<typename T , unsigned int N>
using OptixCoopVec< T, N >::value_type = T
```

### 7.26.3 Constructor & Destructor Documentation

#### 7.26.3.1 OptixCoopVec() [1/2]

```
template<typename T , unsigned int N>
__forceinline__ __device__ OptixCoopVec< T, N >::OptixCoopVec () [inline]
```

#### 7.26.3.2 OptixCoopVec() [2/2]

```
template<typename T , unsigned int N>
__forceinline__ __device__ OptixCoopVec< T, N >::OptixCoopVec (
 const value_type & val) [inline]
```

### 7.26.4 Member Function Documentation

#### 7.26.4.1 data() [1/2]

```
template<typename T , unsigned int N>
__forceinline__ __device__ value_type * OptixCoopVec< T, N >::data ()
[inline]
```

#### 7.26.4.2 data() [2/2]

```
template<typename T , unsigned int N>
__forceinline__ __device__ const value_type * OptixCoopVec< T, N >::data (
) const [inline]
```

#### 7.26.4.3 operator[]() [1/2]

```
template<typename T , unsigned int N>
```

```
__forceinline__ __device__ value_type & OptixCoopVec< T, N >::operator[] (
 unsigned int index) [inline]
```

#### 7.26.4.4 operator[]() [2/2]

```
template<typename T , unsigned int N>
__forceinline__ __device__ const value_type & OptixCoopVec< T, N >
::operator[] (
 unsigned int index) const [inline]
```

### 7.26.5 Member Data Documentation

#### 7.26.5.1 m\_data

```
template<typename T , unsigned int N>
value_type OptixCoopVec< T, N >::m_data[size] [protected]
```

#### 7.26.5.2 size

```
template<typename T , unsigned int N>
const unsigned int OptixCoopVec< T, N >::size = N [static]
```

## 7.27 OptixCoopVecMatrixDescription Struct Reference

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

### Public Attributes

- unsigned int N
- unsigned int K
- unsigned int offsetInBytes
- OptixCoopVecElemType elementType
- OptixCoopVecMatrixLayout layout
- unsigned int rowColumnStrideInBytes
- unsigned int sizeInBytes

#### 7.27.1 Detailed Description

Each matrix's offset from the base address is expressed with offsetInBytes. This allows for non-uniform matrices to be tightly packed.

The rowColumnStrideInBytes is ignored if the layout is either OPTIX\_COOP\_VEC\_MATRIX\_LAYOUT\_INTERENCING\_OPTIMAL or OPTIX\_COOP\_VEC\_MATRIX\_LAYOUT\_TRAINING\_OPTIMAL

### 7.27.2 Member Data Documentation

#### 7.27.2.1 elementType

```
OptixCoopVecElemType OptixCoopVecMatrixDescription::elementType
```

#### 7.27.2.2 K

```
unsigned int OptixCoopVecMatrixDescription::K
```

### 7.27.2.3 layout

[OptixCoopVecMatrixLayout](#) [OptixCoopVecMatrixDescription::layout](#)

### 7.27.2.4 N

unsigned int [OptixCoopVecMatrixDescription::N](#)

### 7.27.2.5 offsetInBytes

unsigned int [OptixCoopVecMatrixDescription::offsetInBytes](#)

### 7.27.2.6 rowColumnStrideInBytes

unsigned int [OptixCoopVecMatrixDescription::rowColumnStrideInBytes](#)

### 7.27.2.7 sizeInBytes

unsigned int [OptixCoopVecMatrixDescription::sizeInBytes](#)

## 7.28 OptixDenoiserGuideLayer Struct Reference

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

### Public Attributes

- [OptixImage2D](#) albedo
- [OptixImage2D](#) normal
- [OptixImage2D](#) flow
- [OptixImage2D](#) previousOutputInternalGuideLayer
- [OptixImage2D](#) outputInternalGuideLayer
- [OptixImage2D](#) flowTrustworthiness

### 7.28.1 Detailed Description

Guide layer for the denoiser.

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

### 7.28.2 Member Data Documentation

#### 7.28.2.1 albedo

[OptixImage2D](#) [OptixDenoiserGuideLayer::albedo](#)

#### 7.28.2.2 flow

[OptixImage2D](#) [OptixDenoiserGuideLayer::flow](#)

#### 7.28.2.3 flowTrustworthiness

[OptixImage2D](#) [OptixDenoiserGuideLayer::flowTrustworthiness](#)

#### 7.28.2.4 normal

[OptixImage2D](#) [OptixDenoiserGuideLayer::normal](#)

### 7.28.2.5 outputInternalGuideLayer

[OptixImage2D](#) [OptixDenoiserGuideLayer::outputInternalGuideLayer](#)

### 7.28.2.6 previousOutputInternalGuideLayer

[OptixImage2D](#) [OptixDenoiserGuideLayer::previousOutputInternalGuideLayer](#)

## 7.29 OptixDenoiserLayer Struct Reference

#include <optix\_types.h>

### Public Attributes

- [OptixImage2D](#) input
- [OptixImage2D](#) previousOutput
- [OptixImage2D](#) output
- [OptixDenoiserAOVType](#) type

### 7.29.1 Detailed Description

Input/Output layers for the denoiser.

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

### 7.29.2 Member Data Documentation

#### 7.29.2.1 input

[OptixImage2D](#) [OptixDenoiserLayer::input](#)

#### 7.29.2.2 output

[OptixImage2D](#) [OptixDenoiserLayer::output](#)

#### 7.29.2.3 previousOutput

[OptixImage2D](#) [OptixDenoiserLayer::previousOutput](#)

#### 7.29.2.4 type

[OptixDenoiserAOVType](#) [OptixDenoiserLayer::type](#)

## 7.30 OptixDenoiserOptions Struct Reference

#include <optix\_types.h>

### Public Attributes

- unsigned int [guideAlbedo](#)
- unsigned int [guideNormal](#)
- [OptixDenoiserAlphaMode](#) denoiseAlpha

### 7.30.1 Detailed Description

Options used by the denoiser.

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

## 7.30.2 Member Data Documentation

### 7.30.2.1 denoiseAlpha

[OptixDenoiserAlphaMode](#) [OptixDenoiserOptions::denoiseAlpha](#)

alpha denoise mode

### 7.30.2.2 guideAlbedo

unsigned int [OptixDenoiserOptions::guideAlbedo](#)

### 7.30.2.3 guideNormal

unsigned int [OptixDenoiserOptions::guideNormal](#)

## 7.31 OptixDenoiserParams Struct Reference

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

### Public Attributes

- [CUdeviceptr](#) [hdrIntensity](#)
- float [blendFactor](#)
- [CUdeviceptr](#) [hdrAverageColor](#)
- unsigned int [temporalModeUsePreviousLayers](#)
- float [flowMulX](#)
- float [flowMulY](#)

### 7.31.1 Detailed Description

Various parameters used by the denoiser.

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

[optixDenoiserComputeIntensity\(\)](#)

[optixDenoiserComputeAverageColor\(\)](#)

## 7.31.2 Member Data Documentation

### 7.31.2.1 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.

### 7.31.2.2 flowMulX

float [OptixDenoiserParams::flowMulX](#)

Multiplication factors for motion vectors (flow guide layer). When set to zero, motion vectors are not scaled.

### 7.31.2.3 flowMulY

float [OptixDenoiserParams::flowMulY](#)

### 7.31.2.4 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.

### 7.31.2.5 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.

### 7.31.2.6 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.

## 7.32 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`

### 7.32.1 Detailed Description

Various sizes related to the denoiser.

See also `optixDenoiserComputeMemoryResources()`

### 7.32.2 Member Data Documentation

#### 7.32.2.1 computeAverageColorSizeInBytes

`size_t` `OptixDenoiserSizes::computeAverageColorSizeInBytes`

Size of scratch memory passed to `optixDenoiserComputeAverageColor`. The size is independent of the tile/image resolution.

#### 7.32.2.2 computeIntensitySizeInBytes

`size_t` `OptixDenoiserSizes::computeIntensitySizeInBytes`

Size of scratch memory passed to [optixDenoiserComputeIntensity](#). The size is independent of the tile/image resolution.

#### 7.32.2.3 internalGuideLayerPixelSizeInBytes

`size_t OptixDenoiserSizes::internalGuideLayerPixelSizeInBytes`

Number of bytes for each pixel in internal guide layers.

#### 7.32.2.4 overlapWindowSizeInPixels

`unsigned int OptixDenoiserSizes::overlapWindowSizeInPixels`

Overlap on all four tile sides.

#### 7.32.2.5 stateSizeInBytes

`size_t OptixDenoiserSizes::stateSizeInBytes`

Size of state memory passed to [optixDenoiserSetup](#), [optixDenoiserInvoke](#).

#### 7.32.2.6 withoutOverlapScratchSizeInBytes

`size_t OptixDenoiserSizes::withoutOverlapScratchSizeInBytes`

Size of scratch memory passed to [optixDenoiserSetup](#), [optixDenoiserInvoke](#). No overlap added.

#### 7.32.2.7 withOverlapScratchSizeInBytes

`size_t OptixDenoiserSizes::withOverlapScratchSizeInBytes`

Size of scratch memory passed to [optixDenoiserSetup](#), [optixDenoiserInvoke](#). Overlap added to dimensions passed to [optixDenoiserComputeMemoryResources](#).

### 7.33 OptixDeviceContextOptions Struct Reference

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

#### Public Attributes

- [OptixLogCallback](#) `logCallbackFunction`
- `void *` `logCallbackData`
- `int` `logCallbackLevel`
- [OptixDeviceContextValidationMode](#) `validationMode`

#### 7.33.1 Detailed Description

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

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

#### 7.33.2 Member Data Documentation

##### 7.33.2.1 logCallbackData

`void* OptixDeviceContextOptions::logCallbackData`

Pointer stored and passed to `logCallbackFunction` when a message is generated.

### 7.33.2.2 logCallbackFunction

[OptixLogCallback](#) [OptixDeviceContextOptions::logCallbackFunction](#)

Function pointer used when OptiX wishes to generate messages.

### 7.33.2.3 logCallbackLevel

`int OptixDeviceContextOptions::logCallbackLevel`

Maximum callback level to generate message for (see [OptixLogCallback](#))

### 7.33.2.4 validationMode

[OptixDeviceContextValidationMode](#) [OptixDeviceContextOptions::validationMode](#)

Validation mode of context.

## 7.34 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(* optixModuleCancelCreation)(OptixModule module, OptixCreationFlags flags)`
- `OptixResult(* optixStub)(void)`
- `OptixResult(* optixDeviceContextCancelCreations)(OptixDeviceContext context, OptixCreationFlags flags)`
- `OptixResult(* optixModuleDestroy)(OptixModule module)`
- `OptixResult(* optixBuiltinISModuleGet)(OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixBuiltinISOptions *builtinISOptions, OptixModule *builtinModule)`

#### Tasks

- `OptixResult(* optixTaskExecute)(OptixTask task, OptixTask *additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int *numAdditionalTasksCreated)`
- `OptixResult(* optixTaskGetSerializationKey)(OptixTask task, void *key, size_t *size)`
- `OptixResult(* optixTaskSerializeOutput)(OptixTask task, void *data, size_t *size)`
- `OptixResult(* optixTaskDeserializeOutput)(OptixTask task, const void *data, size_t size, 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(* optixPipelineSetStackSizeFromCallDepths)(OptixPipeline pipeline, unsigned int maxTraceDepth, unsigned int maxContinuationCallableDepth, unsigned int maxDirectCallableDepthFromState, unsigned int maxDirectCallableDepthFromTraversal, unsigned int maxTraversableGraphDepth)`
- `OptixResult(* optixPipelineSetStackSize)(OptixPipeline pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)`
- `OptixResult(* optixPipelineSymbolMemcpyAsync)(OptixPipeline pipeline, const char *name, void *mem, size_t sizeInBytes, size_t offsetInBytes, OptixPipelineSymbolMemcpyKind kind, CUstream stream)`

#### 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(* stub1)(void)`
- `OptixResult(* stub2)(void)`
- `OptixResult(* optixClusterAccelComputeMemoryUsage)(OptixDeviceContext context, OptixClusterAccelBuildMode buildMode, const OptixClusterAccelBuildInput *buildInput, OptixAccelBufferSizes *bufferSizes)`
- `OptixResult(* optixClusterAccelBuild)(OptixDeviceContext context, CUstream stream, const OptixClusterAccelBuildModeDesc *buildModeDesc, const OptixClusterAccelBuildInput *buildInput, CUdeviceptr argsArray, CUdeviceptr argsCount, unsigned int argsStrideInBytes)`

#### 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)`

#### Cooperative Vector

- `OptixResult(* optixCoopVecMatrixConvert)(OptixDeviceContext context, CUstream stream, unsigned int numNetworks, const OptixNetworkDescription *inputNetworkDescription, CUdeviceptr inputNetworks, size_t inputNetworkStrideInBytes, const OptixNetworkDescription *outputNetworkDescription, CUdeviceptr outputNetworks, size_t outputNetworkStrideInBytes)`

- `OptixResult(* optixCoopVecMatrixComputeSize)(OptixDeviceContext context, unsigned int N, unsigned int K, OptixCoopVecElemType elementType, OptixCoopVecMatrixLayout layout, size_t rowColumnStrideInBytes, size_t *sizeInBytes)`

#### 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)`

### 7.34.1 Detailed Description

The function table containing all API functions.

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

### 7.34.2 Member Data Documentation

#### 7.34.2.1 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()`.

#### 7.34.2.2 optixAccelCompact

`OptixResult(* OptixFunctionTable::optixAccelCompact) (OptixDeviceContext context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle)`

See `optixAccelCompact()`.

### 7.34.2.3 optixAccelComputeMemoryUsage

```
OptixResult(* OptixFunctionTable::optixAccelComputeMemoryUsage)
(OptixDeviceContext context, const OptixAccelBuildOptions *accelOptions,
const OptixBuildInput *buildInputs, unsigned int numBuildInputs,
OptixAccelBufferSizes *bufferSizes)
```

See `optixAccelComputeMemoryUsage()`.

### 7.34.2.4 optixAccelEmitProperty

```
OptixResult(* OptixFunctionTable::optixAccelEmitProperty)
(OptixDeviceContext context, CUstream stream, OptixTraversableHandle handle,
const OptixAccelEmitDesc *emittedProperty)
```

See `optixAccelComputeMemoryUsage()`.

### 7.34.2.5 optixAccelGetRelocationInfo

```
OptixResult(* OptixFunctionTable::optixAccelGetRelocationInfo)
(OptixDeviceContext context, OptixTraversableHandle handle,
OptixRelocationInfo *info)
```

See `optixAccelGetRelocationInfo()`.

### 7.34.2.6 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()`.

### 7.34.2.7 optixBuiltinISModuleGet

```
OptixResult(* OptixFunctionTable::optixBuiltinISModuleGet)
(OptixDeviceContext context, const OptixModuleCompileOptions
*moduleCompileOptions, const OptixPipelineCompileOptions
*pipelineCompileOptions, const OptixBuiltinISOptions *builtinISOptions,
OptixModule *builtinModule)
```

See `optixBuiltinISModuleGet()`.

### 7.34.2.8 optixCheckRelocationCompatibility

```
OptixResult(* OptixFunctionTable::optixCheckRelocationCompatibility)
(OptixDeviceContext context, const OptixRelocationInfo *info, int
*compatible)
```

See `optixCheckRelocationCompatibility()`.

### 7.34.2.9 optixClusterAccelBuild

```
OptixResult(* OptixFunctionTable::optixClusterAccelBuild)
(OptixDeviceContext context, CUstream stream, const
OptixClusterAccelBuildModeDesc *buildModeDesc, const
OptixClusterAccelBuildInput *buildInput, CUdeviceptr argsArray, CUdeviceptr
```

argsCount, unsigned int argsStrideInBytes)

See `optixClusterAccelBuild()`.

#### 7.34.2.10 `optixClusterAccelComputeMemoryUsage`

`OptixResult(* OptixFunctionTable::optixClusterAccelComputeMemoryUsage)`  
 (`OptixDeviceContext` context, `OptixClusterAccelBuildMode` buildMode, const  
`OptixClusterAccelBuildInput` \*buildInput, `OptixAccelBufferSizes` \*bufferSizes)

See `optixClusterAccelComputeMemoryUsage()`.

#### 7.34.2.11 `optixConvertPointerToTraversableHandle`

`OptixResult(* OptixFunctionTable::optixConvertPointerToTraversableHandle)`  
 (`OptixDeviceContext` onDevice, `CUdeviceptr` pointer, `OptixTraversableType`  
 traversableType, `OptixTraversableHandle` \*traversableHandle)

See `optixConvertPointerToTraversableHandle()`.

#### 7.34.2.12 `optixCoopVecMatrixComputeSize`

`OptixResult(* OptixFunctionTable::optixCoopVecMatrixComputeSize)`  
 (`OptixDeviceContext` context, unsigned int N, unsigned int K,  
`OptixCoopVecElemType` elementType, `OptixCoopVecMatrixLayout` layout, size\_t  
 rowColumnStrideInBytes, size\_t \*sizeInBytes)

See `optixCoopVecMatrixComputeSize()`.

#### 7.34.2.13 `optixCoopVecMatrixConvert`

`OptixResult(* OptixFunctionTable::optixCoopVecMatrixConvert)`  
 (`OptixDeviceContext` context, `CUstream` stream, unsigned int numNetworks,  
 const `OptixNetworkDescription` \*inputNetworkDescription, `CUdeviceptr`  
 inputNetworks, size\_t inputNetworkStrideInBytes, const  
`OptixNetworkDescription` \*outputNetworkDescription, `CUdeviceptr`  
 outputNetworks, size\_t outputNetworkStrideInBytes)

See `optixCoopVecMatrixConvert()`.

#### 7.34.2.14 `optixDenoiserComputeAverageColor`

`OptixResult(* OptixFunctionTable::optixDenoiserComputeAverageColor)`  
 (`OptixDenoiser` handle, `CUstream` stream, const `OptixImage2D` \*inputImage,  
`CUdeviceptr` outputAverageColor, `CUdeviceptr` scratch, size\_t  
 scratchSizeInBytes)

See `optixDenoiserComputeAverageColor()`.

#### 7.34.2.15 `optixDenoiserComputeIntensity`

`OptixResult(* OptixFunctionTable::optixDenoiserComputeIntensity)`  
 (`OptixDenoiser` handle, `CUstream` stream, const `OptixImage2D` \*inputImage,  
`CUdeviceptr` outputIntensity, `CUdeviceptr` scratch, size\_t scratchSizeInBytes)

See `optixDenoiserComputeIntensity()`.

### 7.34.2.16 optixDenoiserComputeMemoryResources

`OptixResult(* OptixFunctionTable::optixDenoiserComputeMemoryResources)`  
 (const `OptixDenoiser` handle, unsigned int maximumInputWidth, unsigned int maximumInputHeight, `OptixDenoiserSizes` \*returnSizes)

See `optixDenoiserComputeMemoryResources()`.

### 7.34.2.17 optixDenoiserCreate

`OptixResult(* OptixFunctionTable::optixDenoiserCreate)` (`OptixDeviceContext` context, `OptixDenoiserModelKind` modelKind, const `OptixDenoiserOptions` \*options, `OptixDenoiser` \*returnHandle)

See `optixDenoiserCreate()`.

### 7.34.2.18 optixDenoiserCreateWithUserModel

`OptixResult(* OptixFunctionTable::optixDenoiserCreateWithUserModel)`  
 (`OptixDeviceContext` context, const void \*data, size\_t dataSizeInBytes, `OptixDenoiser` \*returnHandle)

See `optixDenoiserCreateWithUserModel()`.

### 7.34.2.19 optixDenoiserDestroy

`OptixResult(* OptixFunctionTable::optixDenoiserDestroy)` (`OptixDenoiser` handle)

See `optixDenoiserDestroy()`.

### 7.34.2.20 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()`.

### 7.34.2.21 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()`.

### 7.34.2.22 optixDeviceContextCancelCreations

`OptixResult(* OptixFunctionTable::optixDeviceContextCancelCreations)`  
 (`OptixDeviceContext` context, `OptixCreationFlags` flags)

See `optixDeviceContextCancelCreations()`.

#### 7.34.2.23 optixDeviceContextCreate

```
OptixResult(* OptixFunctionTable::optixDeviceContextCreate) (CUcontext
fromContext, const OptixDeviceContextOptions *options, OptixDeviceContext
*context)
```

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

#### 7.34.2.24 optixDeviceContextDestroy

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

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

#### 7.34.2.25 optixDeviceContextGetCacheDatabaseSizes

```
OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheDatabaseSizes)
(OptixDeviceContext context, size_t *lowWaterMark, size_t *highWaterMark)
```

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

#### 7.34.2.26 optixDeviceContextGetCacheEnabled

```
OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheEnabled)
(OptixDeviceContext context, int *enabled)
```

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

#### 7.34.2.27 optixDeviceContextGetCacheLocation

```
OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheLocation)
(OptixDeviceContext context, char *location, size_t locationSize)
```

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

#### 7.34.2.28 optixDeviceContextGetProperty

```
OptixResult(* OptixFunctionTable::optixDeviceContextGetProperty)
(OptixDeviceContext context, OptixDeviceProperty property, void *value, size
_t sizeInBytes)
```

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

#### 7.34.2.29 optixDeviceContextSetCacheDatabaseSizes

```
OptixResult(* OptixFunctionTable::optixDeviceContextSetCacheDatabaseSizes)
(OptixDeviceContext context, size_t lowWaterMark, size_t highWaterMark)
```

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

#### 7.34.2.30 optixDeviceContextSetCacheEnabled

```
OptixResult(* OptixFunctionTable::optixDeviceContextSetCacheEnabled)
(OptixDeviceContext context, int enabled)
```

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

## 7.34.2.31 optixDeviceContextSetCacheLocation

```
OptixResult(* OptixFunctionTable::optixDeviceContextSetCacheLocation)
(OptixDeviceContext context, const char *location)
```

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

## 7.34.2.32 optixDeviceContextSetLogCallback

```
OptixResult(* OptixFunctionTable::optixDeviceContextSetLogCallback)
(OptixDeviceContext context, OptixLogCallback callbackFunction, void
*callbackData, unsigned int callbackLevel)
```

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

## 7.34.2.33 optixGetErrorName

```
const char *(* OptixFunctionTable::optixGetErrorName) (OptixResult result)
```

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

## 7.34.2.34 optixGetErrorString

```
const char *(* OptixFunctionTable::optixGetErrorString) (OptixResult result)
```

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

## 7.34.2.35 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\(\)](#).

## 7.34.2.36 optixModuleCancelCreation

```
OptixResult(* OptixFunctionTable::optixModuleCancelCreation) (OptixModule
module, OptixCreationFlags flags)
```

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

## 7.34.2.37 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\(\)](#).

## 7.34.2.38 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()`.

#### 7.34.2.39 `optixModuleDestroy`

`OptixResult(* OptixFunctionTable::optixModuleDestroy) (OptixModule module)`

See `optixModuleDestroy()`.

#### 7.34.2.40 `optixModuleGetCompilationState`

`OptixResult(* OptixFunctionTable::optixModuleGetCompilationState)  
(OptixModule module, OptixModuleCompileState *state)`

See `optixModuleGetCompilationState()`.

#### 7.34.2.41 `optixOpacityMicromapArrayBuild`

`OptixResult(* OptixFunctionTable::optixOpacityMicromapArrayBuild)  
(OptixDeviceContext context, CUstream stream, const  
OptixOpacityMicromapArrayBuildInput *buildInput, const OptixMicromapBuffers  
*buffers)`

See `optixOpacityMicromapArrayBuild()`.

#### 7.34.2.42 `optixOpacityMicromapArrayComputeMemoryUsage`

`OptixResult(* OptixFunctionTable  
::optixOpacityMicromapArrayComputeMemoryUsage) (OptixDeviceContext context,  
const OptixOpacityMicromapArrayBuildInput *buildInput,  
OptixMicromapBufferSizes *bufferSizes)`

See `optixOpacityMicromapArrayComputeMemoryUsage()`.

#### 7.34.2.43 `optixOpacityMicromapArrayGetRelocationInfo`

`OptixResult(* OptixFunctionTable  
::optixOpacityMicromapArrayGetRelocationInfo) (OptixDeviceContext context,  
CUdeviceptr opacityMicromapArray, OptixRelocationInfo *info)`

See `optixOpacityMicromapArrayGetRelocationInfo()`.

#### 7.34.2.44 `optixOpacityMicromapArrayRelocate`

`OptixResult(* OptixFunctionTable::optixOpacityMicromapArrayRelocate)  
(OptixDeviceContext context, CUstream stream, const OptixRelocationInfo  
*info, CUdeviceptr targetOpacityMicromapArray, size_t  
targetOpacityMicromapArraySizeInBytes)`

See `optixOpacityMicromapArrayRelocate()`.

#### 7.34.2.45 `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()`.

#### 7.34.2.46 `optixPipelineDestroy`

`OptixResult(* OptixFunctionTable::optixPipelineDestroy) (OptixPipeline pipeline)`

See `optixPipelineDestroy()`.

#### 7.34.2.47 `optixPipelineSetStackSize`

`OptixResult(* OptixFunctionTable::optixPipelineSetStackSize) (OptixPipeline pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)`

See `optixPipelineSetStackSize()`.

#### 7.34.2.48 `optixPipelineSetStackSizeFromCallDepths`

`OptixResult(* OptixFunctionTable::optixPipelineSetStackSizeFromCallDepths) (OptixPipeline pipeline, unsigned int maxTraceDepth, unsigned int maxContinuationCallableDepth, unsigned int maxDirectCallableDepthFromState, unsigned int maxDirectCallableDepthFromTraversal, unsigned int maxTraversableGraphDepth)`

See `optixPipelineSetStackSizeFromCallDepths()`.

#### 7.34.2.49 `optixPipelineSymbolMemcpyAsync`

`OptixResult(* OptixFunctionTable::optixPipelineSymbolMemcpyAsync) (OptixPipeline pipeline, const char *name, void *mem, size_t sizeInBytes, size_t offsetInBytes, OptixPipelineSymbolMemcpyKind kind, CUstream stream)`

See `optixPipelineCreate()`.

#### 7.34.2.50 `optixProgramGroupCreate`

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

See `optixProgramGroupCreate()`.

#### 7.34.2.51 `optixProgramGroupDestroy`

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

See `optixProgramGroupDestroy()`.

#### 7.34.2.52 `optixProgramGroupGetStackSize`

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

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

#### 7.34.2.53 [optixSbtRecordPackHeader](#)

[OptixResult](#)(\* [OptixFunctionTable::optixSbtRecordPackHeader](#)) ([OptixProgramGroup](#) programGroup, void \*sbtRecordHeaderHostPointer)

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

#### 7.34.2.54 [optixStub](#)

[OptixResult](#)(\* [OptixFunctionTable::optixStub](#)) (void)

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

#### 7.34.2.55 [optixTaskDeserializeOutput](#)

[OptixResult](#)(\* [OptixFunctionTable::optixTaskDeserializeOutput](#)) ([OptixTask](#) task, const void \*data, size\_t size, [OptixTask](#) \*additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int \*numAdditionalTasksCreated)

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

#### 7.34.2.56 [optixTaskExecute](#)

[OptixResult](#)(\* [OptixFunctionTable::optixTaskExecute](#)) ([OptixTask](#) task, [OptixTask](#) \*additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int \*numAdditionalTasksCreated)

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

#### 7.34.2.57 [optixTaskGetSerializationKey](#)

[OptixResult](#)(\* [OptixFunctionTable::optixTaskGetSerializationKey](#)) ([OptixTask](#) task, void \*key, size\_t \*size)

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

#### 7.34.2.58 [optixTaskSerializeOutput](#)

[OptixResult](#)(\* [OptixFunctionTable::optixTaskSerializeOutput](#)) ([OptixTask](#) task, void \*data, size\_t \*size)

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

#### 7.34.2.59 [stub1](#)

[OptixResult](#)(\* [OptixFunctionTable::stub1](#)) (void)

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

#### 7.34.2.60 [stub2](#)

[OptixResult](#)(\* [OptixFunctionTable::stub2](#)) (void)

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

### 7.35 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

### 7.35.1 Detailed Description

Image descriptor used by the denoiser.

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

### 7.35.2 Member Data Documentation

#### 7.35.2.1 data

[CUdeviceptr](#) [OptixImage2D::data](#)

Pointer to the actual pixel data.

#### 7.35.2.2 format

[OptixPixelFormat](#) [OptixImage2D::format](#)

Pixel format.

#### 7.35.2.3 height

unsigned int [OptixImage2D::height](#)

Height of the image (in pixels)

#### 7.35.2.4 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](#).

#### 7.35.2.5 rowStrideInBytes

unsigned int [OptixImage2D::rowStrideInBytes](#)

Stride between subsequent rows of the image (in bytes).

#### 7.35.2.6 width

unsigned int [OptixImage2D::width](#)

Width of the image (in pixels)

## 7.36 OptixIncomingHitObject Struct Reference

```
#include <optix_device.h>
```

## Public Member Functions

- `__forceinline__ __device__ float getRayTime () const`
- `__forceinline__ __device__ unsigned int getTransformListSize () const`
- `__forceinline__ __device__ OptixTraversableHandle getTransformListHandle (unsigned int index) const`

## 7.36.1 Member Function Documentation

### 7.36.1.1 getRayTime()

```
__forceinline__ __device__ float OptixIncomingHitObject::getRayTime ()
const [inline]
```

### 7.36.1.2 getTransformListHandle()

```
__forceinline__ __device__ OptixTraversableHandle OptixIncomingHitObject
::getTransformListHandle (
 unsigned int index) const [inline]
```

### 7.36.1.3 getTransformListSize()

```
__forceinline__ __device__ unsigned int OptixIncomingHitObject
::getTransformListSize () const [inline]
```

## 7.37 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]

## 7.37.1 Detailed Description

Instances.

See also `OptixBuildInputInstanceArray::instances`

## 7.37.2 Member Data Documentation

### 7.37.2.1 flags

```
unsigned int OptixInstance::flags
```

Any combination of `OptixInstanceFlags` is allowed.

### 7.37.2.2 instanceId

```
unsigned int OptixInstance::instanceId
```

Application supplied ID. The maximal ID can be queried using `OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID`.

### 7.37.2.3 pad

`unsigned int OptixInstance::pad[2]`

round up to 80-byte, to ensure 16-byte alignment

### 7.37.2.4 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`.

### 7.37.2.5 transform

`float OptixInstance::transform[12]`

affine object-to-world transformation as 3x4 matrix in row-major layout

### 7.37.2.6 traversableHandle

`OptixTraversableHandle OptixInstance::traversableHandle`

Set with an `OptixTraversableHandle`.

### 7.37.2.7 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`.

## 7.38 OptixMatrixMotionTransform Struct Reference

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

### Public Attributes

- `OptixTraversableHandle child`
- `OptixMotionOptions motionOptions`
- `unsigned int pad [3]`
- `float transform [2][12]`

### 7.38.1 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* matrixMotionTransform = (OptixMatrixMotionTransform*)
malloc(transformSizeInBytes);
memset(matrixMotionTransform, 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\(\)](#)

## 7.38.2 Member Data Documentation

### 7.38.2.1 child

[OptixTraversableHandle](#) [OptixMatrixMotionTransform::child](#)

The traversable that is transformed by this transformation.

### 7.38.2.2 motionOptions

[OptixMotionOptions](#) [OptixMatrixMotionTransform::motionOptions](#)

The motion options for this transformation. Must have at least two motion keys.

### 7.38.2.3 pad

`unsigned int` [OptixMatrixMotionTransform::pad\[3\]](#)

Padding to make the transformation 16 byte aligned.

### 7.38.2.4 transform

`float` [OptixMatrixMotionTransform::transform\[2\]\[12\]](#)

Affine object-to-world transformation as 3x4 matrix in row-major layout.

## 7.39 OptixMicromapBuffers Struct Reference

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

### Public Attributes

- [CUdeviceptr](#) [output](#)
- [size\\_t](#) [outputSizeInBytes](#)
- [CUdeviceptr](#) [temp](#)
- [size\\_t](#) [tempSizeInBytes](#)

### 7.39.1 Detailed Description

Buffer inputs for opacity micromap array builds.

## 7.39.2 Member Data Documentation

### 7.39.2.1 output

[CUdeviceptr](#) [OptixMicromapBuffers::output](#)

Output buffer.

### 7.39.2.2 outputSizeInBytes

[size\\_t](#) [OptixMicromapBuffers::outputSizeInBytes](#)

Output buffer size.

### 7.39.2.3 temp

`CUdeviceptr` `OptixMicromapBuffers::temp`

Temp buffer.

### 7.39.2.4 tempSizeInBytes

`size_t` `OptixMicromapBuffers::tempSizeInBytes`

Temp buffer size.

## 7.40 OptixMicromapBufferSizes Struct Reference

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

### Public Attributes

- `size_t` `outputSizeInBytes`
- `size_t` `tempSizeInBytes`

### 7.40.1 Detailed Description

Conservative memory requirements for building a opacity micromap array.

### 7.40.2 Member Data Documentation

#### 7.40.2.1 outputSizeInBytes

`size_t` `OptixMicromapBufferSizes::outputSizeInBytes`

#### 7.40.2.2 tempSizeInBytes

`size_t` `OptixMicromapBufferSizes::tempSizeInBytes`

## 7.41 OptixModuleCompileBoundValueEntry Struct Reference

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

### Public Attributes

- `size_t` `pipelineParamOffsetInBytes`
- `size_t` `sizeInBytes`
- `const void *` `boundValuePtr`
- `const char *` `annotation`

### 7.41.1 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](#)

## 7.41.2 Member Data Documentation

### 7.41.2.1 annotation

```
const char* OptixModuleCompileBoundValueEntry::annotation
```

### 7.41.2.2 boundValuePtr

```
const void* OptixModuleCompileBoundValueEntry::boundValuePtr
```

### 7.41.2.3 pipelineParamOffsetInBytes

```
size_t OptixModuleCompileBoundValueEntry::pipelineParamOffsetInBytes
```

### 7.41.2.4 sizeInBytes

```
size_t OptixModuleCompileBoundValueEntry::sizeInBytes
```

## 7.42 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](#)
- [OptixModule baseModule](#)

### 7.42.1 Detailed Description

Compilation options for module.

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

## 7.42.2 Member Data Documentation

### 7.42.2.1 baseModule

[OptixModule](#) [OptixModuleCompileOptions::baseModule](#)

If not nullptr, pointer to the base module for potential specialization.

### 7.42.2.2 boundValues

const [OptixModuleCompileBoundValueEntry\\*](#) [OptixModuleCompileOptions::boundValues](#)

Ingored if numBoundValues is set to 0.

### 7.42.2.3 debugLevel

[OptixCompileDebugLevel](#) [OptixModuleCompileOptions::debugLevel](#)

Generate debug information.

### 7.42.2.4 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.

### 7.42.2.5 numBoundValues

unsigned int [OptixModuleCompileOptions::numBoundValues](#)

set to 0 if unused

### 7.42.2.6 numPayloadTypes

unsigned int [OptixModuleCompileOptions::numPayloadTypes](#)

The number of different payload types available for compilation. Must be zero if [OptixPipelineCompileOptions::numPayloadValues](#) is not zero.

### 7.42.2.7 optLevel

[OptixCompileOptimizationLevel](#) [OptixModuleCompileOptions::optLevel](#)

Optimization level. May vary within a pipeline.

### 7.42.2.8 payloadTypes

const [OptixPayloadType\\*](#) [OptixModuleCompileOptions::payloadTypes](#)

Points to host array of payload type definitions, size must match numPayloadTypes.

## 7.43 OptixMotionOptions Struct Reference

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

## Public Attributes

- unsigned short `numKeys`
- unsigned short `flags`
- float `timeBegin`
- float `timeEnd`

### 7.43.1 Detailed Description

Motion options.

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

### 7.43.2 Member Data Documentation

#### 7.43.2.1 flags

unsigned short `OptixMotionOptions::flags`

Combinations of `OptixMotionFlags`.

#### 7.43.2.2 numKeys

unsigned short `OptixMotionOptions::numKeys`

If `numKeys > 1`, motion is enabled. `timeBegin`, `timeEnd` and `flags` are all ignored when motion is disabled.

#### 7.43.2.3 timeBegin

float `OptixMotionOptions::timeBegin`

Point in time where motion starts. Must be lesser than `timeEnd`.

#### 7.43.2.4 timeEnd

float `OptixMotionOptions::timeEnd`

Point in time where motion ends. Must be greater than `timeBegin`.

## 7.44 OptixNetworkDescription Struct Reference

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

## Public Attributes

- `OptixCoopVecMatrixDescription * layers`
- unsigned int `numLayers`

### 7.44.1 Member Data Documentation

#### 7.44.1.1 layers

`OptixCoopVecMatrixDescription*` `OptixNetworkDescription::layers`

#### 7.44.1.2 numLayers

unsigned int `OptixNetworkDescription::numLayers`

## 7.45 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`

### 7.45.1 Detailed Description

Inputs to opacity micromap array construction.

### 7.45.2 Member Data Documentation

#### 7.45.2.1 `flags`

```
unsigned int OptixOpacityMicromapArrayBuildInput::flags
```

Applies to all opacity micromaps in array.

#### 7.45.2.2 `inputBuffer`

```
CUdeviceptr OptixOpacityMicromapArrayBuildInput::inputBuffer
```

128B aligned base pointer for raw opacity micromap input data.

#### 7.45.2.3 `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.

#### 7.45.2.4 `numMicromapHistogramEntries`

```
unsigned int OptixOpacityMicromapArrayBuildInput
::numMicromapHistogramEntries
```

Number of `OptixOpacityMicromapHistogramEntry`.

#### 7.45.2.5 `perMicromapDescBuffer`

```
CUdeviceptr OptixOpacityMicromapArrayBuildInput::perMicromapDescBuffer
```

One `OptixOpacityMicromapDesc` entry per opacity micromap. This device pointer must be a multiple of `OPTIX_OPACITY_MICROMAP_DESC_BYTE_ALIGNMENT`.

#### 7.45.2.6 `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.

## 7.46 OptixOpacityMicromapDesc Struct Reference

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

### Public Attributes

- unsigned int [byteOffset](#)
- unsigned short [subdivisionLevel](#)
- unsigned short [format](#)

### 7.46.1 Detailed Description

Opacity micromap descriptor.

### 7.46.2 Member Data Documentation

#### 7.46.2.1 byteOffset

unsigned int OptixOpacityMicromapDesc::byteOffset

Byte offset to opacity micromap in data input buffer of opacity micromap array build.

#### 7.46.2.2 format

unsigned short OptixOpacityMicromapDesc::format

OptixOpacityMicromapFormat.

#### 7.46.2.3 subdivisionLevel

unsigned short OptixOpacityMicromapDesc::subdivisionLevel

Number of micro-triangles is  $4^{\text{level}}$ . Valid levels are [0, 12].

## 7.47 OptixOpacityMicromapHistogramEntry Struct Reference

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

### Public Attributes

- unsigned int [count](#)
- unsigned int [subdivisionLevel](#)
- [OptixOpacityMicromapFormat](#) format

### 7.47.1 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.

### 7.47.2 Member Data Documentation

#### 7.47.2.1 count

unsigned int OptixOpacityMicromapHistogramEntry::count

Number of opacity micromaps with the format and subdivision level that are input to the opacity micromap array build.

#### 7.47.2.2 format

`OptixOpacityMicromapFormat` `OptixOpacityMicromapHistogramEntry::format`

Opacity micromap format.

#### 7.47.2.3 subdivisionLevel

`unsigned int` `OptixOpacityMicromapHistogramEntry::subdivisionLevel`

Number of micro-triangles is  $4^{\text{level}}$ . Valid levels are [0, 12].

### 7.48 OptixOpacityMicromapUsageCount Struct Reference

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

#### Public Attributes

- `unsigned int` `count`
- `unsigned int` `subdivisionLevel`
- `OptixOpacityMicromapFormat` `format`

#### 7.48.1 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.

#### 7.48.2 Member Data Documentation

##### 7.48.2.1 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.

##### 7.48.2.2 format

`OptixOpacityMicromapFormat` `OptixOpacityMicromapUsageCount::format`

opacity micromap format.

##### 7.48.2.3 subdivisionLevel

`unsigned int` `OptixOpacityMicromapUsageCount::subdivisionLevel`

Number of micro-triangles is  $4^{\text{level}}$ . Valid levels are [0, 12].

### 7.49 OptixOutgoingHitObject Struct Reference

```
#include <optix_device.h>
```

#### Public Member Functions

- `__forceinline__ __device__ float getRayTime () const`
- `__forceinline__ __device__ unsigned int getTransformListSize () const`
- `__forceinline__ __device__ OptixTraversableHandle getTransformListHandle (unsigned int index) const`

## 7.49.1 Member Function Documentation

### 7.49.1.1 getRayTime()

```
__forceinline__ __device__ float OptixOutgoingHitObject::getRayTime ()
const [inline]
```

### 7.49.1.2 getTransformListHandle()

```
__forceinline__ __device__ OptixTraversableHandle OptixOutgoingHitObject
::getTransformListHandle (
 unsigned int index) const [inline]
```

### 7.49.1.3 getTransformListSize()

```
__forceinline__ __device__ unsigned int OptixOutgoingHitObject
::getTransformListSize () const [inline]
```

## 7.50 OptixPayloadType Struct Reference

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

### Public Attributes

- unsigned int `numPayloadValues`
- const unsigned int \* `payloadSemantics`

## 7.50.1 Detailed Description

Specifies a single payload type.

## 7.50.2 Member Data Documentation

### 7.50.2.1 numPayloadValues

```
unsigned int OptixPayloadType::numPayloadValues
```

The number of 32b words the payload of this type holds.

### 7.50.2.2 payloadSemantics

```
const unsigned int* OptixPayloadType::payloadSemantics
```

Points to host array of payload word semantics, size must match numPayloadValues.

## 7.51 OptixPipelineCompileOptions Struct Reference

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

### Public Attributes

- int `usesMotionBlur`

- unsigned int traversableGraphFlags
- int numPayloadValues
- int numAttributeValues
- unsigned int exceptionFlags
- const char \* pipelineLaunchParamsVariableName
- size\_t pipelineLaunchParamsSizeInBytes
- unsigned int usesPrimitiveTypeFlags
- int allowOpacityMicromaps
- int allowClusteredGeometry

### 7.51.1 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\(\)](#)

### 7.51.2 Member Data Documentation

#### 7.51.2.1 allowClusteredGeometry

`int OptixPipelineCompileOptions::allowClusteredGeometry`

Boolean value indicating whether clusters (cluster acceleration structures) may be used. This value MUST be set if clusters are present in the BVH, otherwise validation will return an error.

#### 7.51.2.2 allowOpacityMicromaps

`int OptixPipelineCompileOptions::allowOpacityMicromaps`

Boolean value indicating whether opacity micromaps may be used.

#### 7.51.2.3 exceptionFlags

`unsigned int OptixPipelineCompileOptions::exceptionFlags`

A bitmask of [OptixExceptionFlags](#) indicating which exceptions are enabled.

#### 7.51.2.4 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].

#### 7.51.2.5 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.

#### 7.51.2.6 pipelineLaunchParamsSizeInBytes

`size_t OptixPipelineCompileOptions::pipelineLaunchParamsSizeInBytes`

Size of the variable pointed to by [pipelineLaunchParamsVariableName](#). It will be a compiler error if the size of the variable pointed to by [pipelineLaunchParamsVariableName](#) is not equal to this size.

### 7.51.2.7 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.

### 7.51.2.8 traversableGraphFlags

`unsigned int OptixPipelineCompileOptions::traversableGraphFlags`

Traversable graph bitfield. See `OptixTraversableGraphFlags`.

### 7.51.2.9 usesMotionBlur

`int OptixPipelineCompileOptions::usesMotionBlur`

Boolean value indicating whether motion blur could be used.

### 7.51.2.10 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`.

## 7.52 OptixPipelineLinkOptions Struct Reference

`#include <optix_types.h>`

### Public Attributes

- unsigned int `maxTraceDepth`
- unsigned int `maxContinuationCallableDepth`
- unsigned int `maxDirectCallableDepthFromState`
- unsigned int `maxDirectCallableDepthFromTraversal`
- unsigned int `maxTraversableGraphDepth`

### 7.52.1 Detailed Description

Link options for a pipeline.

See also `optixPipelineCreate()`

### 7.52.2 Member Data Documentation

#### 7.52.2.1 maxContinuationCallableDepth

`unsigned int OptixPipelineLinkOptions::maxContinuationCallableDepth`

Maximum depth of continuation callable call graphs. 0 means that continuation callables will not take part in the stack size calculation and can most likely not be called.

#### 7.52.2.2 maxDirectCallableDepthFromState

`unsigned int OptixPipelineLinkOptions::maxDirectCallableDepthFromState`

Maximum depth of direct callable call graphs called from `raygen`, `closesthit`, `miss` or `continuation`

callable programs. 0 means that direct callables will not take part in the default stack size calculation for that part of the pipeline and can not be called from the programs mentioned above if the callable needs any stack.

### 7.52.2.3 maxDirectCallableDepthFromTraversal

```
unsigned int OptixPipelineLinkOptions::maxDirectCallableDepthFromTraversal
```

Maximum depth of direct callable call graphs called from intersect or anyhit programs. 0 means that direct callables will not take part in the default stack size calculation for that part of the pipeline and can not be called from the programs mentioned above if the callable needs any stack.

### 7.52.2.4 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.

### 7.52.2.5 maxTraversableGraphDepth

```
unsigned int OptixPipelineLinkOptions::maxTraversableGraphDepth
```

The maximum depth of a traversable graph passed to trace. 0 means to take a default value based on the traversableGraphFlags passed to [OptixPipelineCompileOptions::traversableGraphFlags](#): OPTIX\_TRAVERSABLE\_GRAPH\_FLAG\_ALLOW\_SINGLE\_GAS means to take 1, otherwise 2 will be taken.

## 7.53 OptixProgramGroupCallables Struct Reference

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

### Public Attributes

- [OptixModule moduleDC](#)
- `const char * entryFunctionNameDC`
- [OptixModule moduleCC](#)
- `const char * entryFunctionNameCC`

### 7.53.1 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](#)

### 7.53.2 Member Data Documentation

#### 7.53.2.1 entryFunctionNameCC

```
const char* OptixProgramGroupCallables::entryFunctionNameCC
```

Entry function name of the continuation callable (CC) program.

#### 7.53.2.2 entryFunctionNameDC

```
const char* OptixProgramGroupCallables::entryFunctionNameDC
```

Entry function name of the direct callable (DC) program.

### 7.53.2.3 moduleCC

[OptixModule](#) [OptixProgramGroupCallables::moduleCC](#)

Module holding the continuation callable (CC) program.

### 7.53.2.4 moduleDC

[OptixModule](#) [OptixProgramGroupCallables::moduleDC](#)

Module holding the direct callable (DC) program.

## 7.54 OptixProgramGroupDesc Struct Reference

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

### Public Attributes

- [OptixProgramGroupKind](#) kind
- unsigned int flags
- union {
  - [OptixProgramGroupHitgroup](#) hitgroup
  - [OptixProgramGroupSingleModule](#) raygen
  - [OptixProgramGroupSingleModule](#) miss
  - [OptixProgramGroupSingleModule](#) exception
  - [OptixProgramGroupCallables](#) callables

### 7.54.1 Detailed Description

Descriptor for program groups.

### 7.54.2 Member Data Documentation

#### 7.54.2.1

```
union { ... } OptixProgramGroupDesc::@9
```

#### 7.54.2.2 callables

[OptixProgramGroupCallables](#) [OptixProgramGroupDesc::callables](#)

See also [OPTIX\\_PROGRAM\\_GROUP\\_KIND\\_CALLABLES](#)

#### 7.54.2.3 exception

[OptixProgramGroupSingleModule](#) [OptixProgramGroupDesc::exception](#)

See also [OPTIX\\_PROGRAM\\_GROUP\\_KIND\\_EXCEPTION](#)

#### 7.54.2.4 flags

unsigned int [OptixProgramGroupDesc::flags](#)

See [OptixProgramGroupFlags](#).

### 7.54.2.5 hitgroup

[OptixProgramGroupHitgroup](#) [OptixProgramGroupDesc::hitgroup](#)

See also [OPTIX\\_PROGRAM\\_GROUP\\_KIND\\_HITGROUP](#)

### 7.54.2.6 kind

[OptixProgramGroupKind](#) [OptixProgramGroupDesc::kind](#)

The kind of program group.

### 7.54.2.7 miss

[OptixProgramGroupSingleModule](#) [OptixProgramGroupDesc::miss](#)

See also [OPTIX\\_PROGRAM\\_GROUP\\_KIND\\_MISS](#)

### 7.54.2.8 raygen

[OptixProgramGroupSingleModule](#) [OptixProgramGroupDesc::raygen](#)

See also [OPTIX\\_PROGRAM\\_GROUP\\_KIND\\_RAYGEN](#)

## 7.55 OptixProgramGroupHitgroup Struct Reference

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

### Public Attributes

- [OptixModule moduleCH](#)
- [const char \\* entryFunctionNameCH](#)
- [OptixModule moduleAH](#)
- [const char \\* entryFunctionNameAH](#)
- [OptixModule moduleIS](#)
- [const char \\* entryFunctionNameIS](#)

### 7.55.1 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](#)

### 7.55.2 Member Data Documentation

#### 7.55.2.1 entryFunctionNameAH

`const char* OptixProgramGroupHitgroup::entryFunctionNameAH`

Entry function name of the any hit (AH) program.

#### 7.55.2.2 entryFunctionNameCH

`const char* OptixProgramGroupHitgroup::entryFunctionNameCH`

Entry function name of the closest hit (CH) program.

### 7.55.2.3 entryFunctionNameIS

`const char* OptixProgramGroupHitgroup::entryFunctionNameIS`

Entry function name of the intersection (IS) program.

### 7.55.2.4 moduleAH

`OptixModule OptixProgramGroupHitgroup::moduleAH`

Module holding the any hit (AH) program.

### 7.55.2.5 moduleCH

`OptixModule OptixProgramGroupHitgroup::moduleCH`

Module holding the closest hit (CH) program.

### 7.55.2.6 moduleIS

`OptixModule OptixProgramGroupHitgroup::moduleIS`

Module holding the intersection (Is) program.

## 7.56 OptixProgramGroupOptions Struct Reference

`#include <optix_types.h>`

### Public Attributes

- `const OptixPayloadType * payloadType`

### 7.56.1 Detailed Description

Program group options.

See also `optixProgramGroupCreate()`

### 7.56.2 Member Data Documentation

#### 7.56.2.1 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`.

## 7.57 OptixProgramGroupSingleModule Struct Reference

`#include <optix_types.h>`

## Public Attributes

- [OptixModule](#) module
- const char \* entryFunctionName

### 7.57.1 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](#)

### 7.57.2 Member Data Documentation

#### 7.57.2.1 entryFunctionName

const char\* OptixProgramGroupSingleModule::entryFunctionName

Entry function name of the single program.

#### 7.57.2.2 module

[OptixModule](#) OptixProgramGroupSingleModule::module

Module holding single program.

## 7.58 OptixRelocateInput Struct Reference

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

## Public Attributes

- [OptixBuildInputType](#) type
  - union {
    - [OptixRelocateInputInstanceArray](#) instanceArray
    - [OptixRelocateInputTriangleArray](#) triangleArray
- ```
};
```

7.58.1 Detailed Description

Relocation inputs.

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

7.58.2 Member Data Documentation

7.58.2.1

union { ... } OptixRelocateInput::@3

7.58.2.2 instanceArray

[OptixRelocateInputInstanceArray](#) OptixRelocateInput::instanceArray

Instance and instance pointer inputs.

7.58.2.3 triangleArray

[OptixRelocateInputTriangleArray](#) [OptixRelocateInput::triangleArray](#)

Triangle inputs.

7.58.2.4 type

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

The type of the build input to relocate.

7.59 OptixRelocateInputInstanceArray Struct Reference

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

Public Attributes

- unsigned int [numInstances](#)
- [CUdeviceptr](#) [traversableHandles](#)

7.59.1 Detailed Description

Instance and instance pointer inputs.

See also [OptixRelocateInput::instanceArray](#)

7.59.2 Member Data Documentation

7.59.2.1 numInstances

unsigned int [OptixRelocateInputInstanceArray::numInstances](#)

Number of elements in [OptixRelocateInputInstanceArray::traversableHandles](#). Must match [OptixBuildInputInstanceArray::numInstances](#) of the source build input.

7.59.2.2 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).

7.60 OptixRelocateInputOpacityMicromap Struct Reference

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

Public Attributes

- [CUdeviceptr](#) [opacityMicromapArray](#)

7.60.1 Member Data Documentation

7.60.1.1 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).

7.61 OptixRelocateInputTriangleArray Struct Reference

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

Public Attributes

- unsigned int [numSbtRecords](#)
- [OptixRelocateInputOpacityMicromap](#) [opacityMicromap](#)

7.61.1 Detailed Description

Triangle inputs.

See also [OptixRelocateInput::triangleArray](#)

7.61.2 Member Data Documentation

7.61.2.1 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.

7.61.2.2 opacityMicromap

```
OptixRelocateInputOpacityMicromap OptixRelocateInputTriangleArray  
::opacityMicromap
```

Opacity micromap inputs.

7.62 OptixRelocationInfo Struct Reference

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

Public Attributes

- unsigned long long [info](#) [4]

7.62.1 Detailed Description

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

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

7.62.2 Member Data Documentation

7.62.2.1 info

```
unsigned long long OptixRelocationInfo::info[4]
```

Opaque data, used internally, should not be modified.

7.63 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`

7.63.1 Detailed Description

Describes the shader binding table (SBT)

See also `optixLaunch()`

7.63.2 Member Data Documentation

7.63.2.1 `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`.

7.63.2.2 `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`.

7.63.2.3 `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`.

7.63.2.4 `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`.

7.63.2.5 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`.

7.63.2.6 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`.

7.63.2.7 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`.

7.63.2.8 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`.

7.63.2.9 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`.

7.63.2.10 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`.

7.63.2.11 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`.

7.64 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 `pvx`
- float `sz`
- float `pvz`
- float `qx`
- float `qy`
- float `qz`
- float `qw`
- float `tx`
- float `ty`
- float `tz`

7.64.1 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\(\)](#)

7.64.2 Member Data Documentation

7.64.2.1 `a`

float `OptixSRTData::a`

7.64.2.2 `b`

float `OptixSRTData::b`

7.64.2.3 `c`

float `OptixSRTData::c`

7.64.2.4 pvx

float OptixSRTData::pvx

7.64.2.5 pvy

float OptixSRTData::pvy

7.64.2.6 pvz

float OptixSRTData::pvz

7.64.2.7 qw

float OptixSRTData::qw

7.64.2.8 qx

float OptixSRTData::qx

7.64.2.9 qy

float OptixSRTData::qy

7.64.2.10 qz

float OptixSRTData::qz

7.64.2.11 sx

float OptixSRTData::sx

7.64.2.12 sy

float OptixSRTData::sy

7.64.2.13 sz

float OptixSRTData::sz

7.64.2.14 tx

float OptixSRTData::tx

7.64.2.15 ty

float OptixSRTData::ty

7.64.2.16 tz

float OptixSRTData::tz

7.65 OptixSRTMotionTransform Struct Reference

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

Public Attributes

- [OptixTraversableHandle](#) child
- [OptixMotionOptions](#) motionOptions
- unsigned int pad [3]
- [OptixSRTData](#) srtData [2]

7.65.1 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\(\)](#)

7.65.2 Member Data Documentation

7.65.2.1 child

[OptixTraversableHandle](#) [OptixSRTMotionTransform::child](#)

The traversable transformed by this transformation.

7.65.2.2 motionOptions

[OptixMotionOptions](#) [OptixSRTMotionTransform::motionOptions](#)

The motion options for this transformation Must have at least two motion keys.

7.65.2.3 pad

unsigned int [OptixSRTMotionTransform::pad](#)[3]

Padding to make the SRT data 16 byte aligned.

7.65.2.4 srtData

[OptixSRTData](#) [OptixSRTMotionTransform::srtData](#)[2]

The actual SRT data describing the transformation.

7.66 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](#)

7.66.1 Detailed Description

Describes the stack size requirements of a program group.

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

7.66.2 Member Data Documentation

7.66.2.1 [cssAH](#)

`unsigned int OptixStackSizes::cssAH`

Continuation stack size of AH programs in bytes.

7.66.2.2 [cssCC](#)

`unsigned int OptixStackSizes::cssCC`

Continuation stack size of CC programs in bytes.

7.66.2.3 [cssCH](#)

`unsigned int OptixStackSizes::cssCH`

Continuation stack size of CH programs in bytes.

7.66.2.4 [cssIS](#)

`unsigned int OptixStackSizes::cssIS`

Continuation stack size of IS programs in bytes.

7.66.2.5 [cssMS](#)

`unsigned int OptixStackSizes::cssMS`

Continuation stack size of MS programs in bytes.

7.66.2.6 [cssRG](#)

`unsigned int OptixStackSizes::cssRG`

Continuation stack size of RG programs in bytes.

7.66.2.7 [dssDC](#)

`unsigned int OptixStackSizes::dssDC`

Direct stack size of DC programs in bytes.

7.67 OptixStaticTransform Struct Reference

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

Public Attributes

- [OptixTraversableHandle](#) child
- unsigned int pad [2]
- float transform [12]
- float invTransform [12]

7.67.1 Detailed Description

Static transform.

The device address of instances of this type must be a multiple of OPTIX_TRANSFORM_BYTE_ALIGNMENT.

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

7.67.2 Member Data Documentation

7.67.2.1 child

[OptixTraversableHandle](#) OptixStaticTransform::child

The traversable transformed by this transformation.

7.67.2.2 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.

7.67.2.3 pad

unsigned int OptixStaticTransform::pad[2]

Padding to make the transformations 16 byte aligned.

7.67.2.4 transform

float OptixStaticTransform::transform[12]

Affine object-to-world transformation as 3x4 matrix in row-major layout.

7.68 OptixTraverseData Struct Reference

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

Public Attributes

- unsigned int data [20]

7.68.1 Detailed Description

Hit Object Struct to store the data collected in a hit object during traversal in an internal format using [optixHitObjectGetTraverseData\(\)](#). The hit object can be reconstructed using that data at a later point with [optixMakeHitObjectWithTraverseData\(\)](#).

7.68.2 Member Data Documentation

7.68.2.1 data

unsigned int OptixTraverseData::data[20]

7.69 OptixUtilDenoiserImageTile Struct Reference

#include <optix_denoiser_tiling.h>

Public Attributes

- [OptixImage2D](#) input
- [OptixImage2D](#) output
- unsigned int [inputOffsetX](#)
- unsigned int [inputOffsetY](#)

7.69.1 Detailed Description

Tile definition.

see [optixUtilDenoiserSplitImage](#)

7.69.2 Member Data Documentation

7.69.2.1 input

[OptixImage2D](#) OptixUtilDenoiserImageTile::input

7.69.2.2 inputOffsetX

unsigned int OptixUtilDenoiserImageTile::inputOffsetX

7.69.2.3 inputOffsetY

unsigned int OptixUtilDenoiserImageTile::inputOffsetY

7.69.2.4 output

[OptixImage2D](#) OptixUtilDenoiserImageTile::output

7.70 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)`
- `static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float rayTime, unsigned int rayFlags, OptixTraverseData traverseData, const OptixTraversableHandle *transforms, unsigned int numTransforms)`
- `static __forceinline__ __device__ void optixMakeMissHitObject (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int rayFlags)`
- `static __forceinline__ __device__ void optixMakeNopHitObject ()`
- `static __forceinline__ __device__ void optixHitObjectGetTraverseData (OptixTraverseData *data)`
- `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__ void optixHitObjectSetSbtRecordIndex (unsigned int sbtRecordIndex)
- static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer ()
- static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetGASTraversableHandle ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetRayFlags ()
- 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 types)
- static __forceinline__ __device__ unsigned int [optixUndefinedValue](#) ()
- __device__ __forceinline__ unsigned int [optixGetRemainingTraceDepth](#) ()
- 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 optixGetTriangleVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])`
- static `__forceinline__ __device__ void optixGetTriangleVertexData (float3 data[3])`
- static `__forceinline__ __device__ void optixHitObjectGetTriangleVertexData (float3 data[3])`
- static `__forceinline__ __device__ void optixGetLinearCurveVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])`
- static `__forceinline__ __device__ void optixGetLinearCurveVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])`
- static `__forceinline__ __device__ void optixGetLinearCurveVertexData (float4 data[2])`
- static `__forceinline__ __device__ void optixHitObjectGetLinearCurveVertexData (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 optixGetQuadraticBSplineVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])`
- static `__forceinline__ __device__ void optixGetQuadraticBSplineVertexData (float4 data[3])`
- static `__forceinline__ __device__ void optixHitObjectGetQuadraticBSplineVertexData (float4 data[3])`
- static `__forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])`
- static `__forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexData (float4 data[3])`
- static `__forceinline__ __device__ void optixHitObjectGetQuadraticBSplineRocapsVertexData (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 optixGetCubicBSplineVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])`
- static `__forceinline__ __device__ void optixGetCubicBSplineVertexData (float4 data[4])`
- static `__forceinline__ __device__ void optixHitObjectGetCubicBSplineVertexData (float4 data[4])`
- static `__forceinline__ __device__ void optixGetCubicBSplineRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])`
- static `__forceinline__ __device__ void optixGetCubicBSplineRocapsVertexData (float4 data[4])`
- static `__forceinline__ __device__ void optixHitObjectGetCubicBSplineRocapsVertexData (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 optixGetCatmullRomVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])`
- static `__forceinline__ __device__ void optixGetCatmullRomVertexData (float4 data[4])`
- static `__forceinline__ __device__ void optixHitObjectGetCatmullRomVertexData (float4 data[4])`

- static __forceinline__ __device__ void [optixGetCatmullRomRocapsVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void [optixGetCatmullRomRocapsVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixHitObjectGetCatmullRomRocapsVertexData](#) (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 [optixGetCubicBezierVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void [optixGetCubicBezierVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixHitObjectGetCubicBezierVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixGetCubicBezierRocapsVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void [optixGetCubicBezierRocapsVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixHitObjectGetCubicBezierRocapsVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixGetRibbonVertexData](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void [optixGetRibbonVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void [optixGetRibbonVertexData](#) (float4 data[3])
- static __forceinline__ __device__ void [optixHitObjectGetRibbonVertexData](#) (float4 data[3])
- static __forceinline__ __device__ float3 [optixGetRibbonNormal](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ float3 [optixGetRibbonNormalFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ float3 [optixGetRibbonNormal](#) (float2 ribbonParameters)
- static __forceinline__ __device__ float3 [optixHitObjectGetRibbonNormal](#) (float2 ribbonParameters)
- static __forceinline__ __device__ void [optixGetSphereData](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ void [optixGetSphereDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ void [optixGetSphereData](#) (float4 data[1])
- static __forceinline__ __device__ void [optixHitObjectGetSphereData](#) (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)
- template<typename HitState >
static __forceinline__ __device__ void [optixGetWorldToObjectTransformMatrix](#) (const HitState &hs, float m[12])
- static __forceinline__ __device__ void [optixGetWorldToObjectTransformMatrix](#) (float m[12])

- static `__forceinline__ __device__ void optixHitObjectGetWorldToObjectTransformMatrix (float m[12])`
- `template<typename HitState >`
static `__forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (const HitState &hs, float m[12])`
- static `__forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (float m[12])`
- static `__forceinline__ __device__ void optixHitObjectGetObjectToWorldTransformMatrix (float m[12])`
- `template<typename HitState >`
static `__forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (const HitState &hs, float3 point)`
- static `__forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (float3 point)`
- static `__forceinline__ __device__ float3 optixHitObjectTransformPointFromWorldToObjectSpace (float3 point)`
- `template<typename HitState >`
static `__forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (const HitState &hs, float3 vec)`
- static `__forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (float3 vec)`
- static `__forceinline__ __device__ float3 optixHitObjectTransformVectorFromWorldToObjectSpace (float3 vec)`
- `template<typename HitState >`
static `__forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (const HitState &hs, float3 normal)`
- static `__forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (float3 normal)`
- static `__forceinline__ __device__ float3`
`optixHitObjectTransformNormalFromWorldToObjectSpace (float3 normal)`
- `template<typename HitState >`
static `__forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (const HitState &hs, float3 point)`
- static `__forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (float3 point)`
- static `__forceinline__ __device__ float3 optixHitObjectTransformPointFromObjectToWorldSpace (float3 point)`
- `template<typename HitState >`
static `__forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (const HitState &hs, float3 vec)`
- static `__forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (float3 vec)`
- static `__forceinline__ __device__ float3 optixHitObjectTransformVectorFromObjectToWorldSpace (float3 vec)`
- `template<typename HitState >`
static `__forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (const HitState &hs, float3 normal)`
- static `__forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (float3 normal)`
- static `__forceinline__ __device__ float3`
`optixHitObjectTransformNormalFromObjectToWorldSpace (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 [optixGetClusterId](#) ()
- static __forceinline__ __device__ unsigned int [optixHitObjectGetClusterId](#) ()
- 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__ float [optixGetCurveParameter](#) ()
- static __forceinline__ __device__ float [optixHitObjectGetCurveParameter](#) ()
- static __forceinline__ __device__ float2 [optixGetRibbonParameters](#) ()
- static __forceinline__ __device__ float2 [optixHitObjectGetRibbonParameters](#) ()
- static __forceinline__ __device__ float2 [optixGetTriangleBarycentrics](#) ()
- static __forceinline__ __device__ float2 [optixHitObjectGetTriangleBarycentrics](#) ()
- 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 optixGetCatmullRomRocapsVertexData()

```
static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexData (
    float4 data[4] ) [static]
```

8.1.3.12 optixGetCatmullRomRocapsVertexDataFromHandle()

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

8.1.3.13 optixGetCatmullRomVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetCatmullRomVertexData (
    float4 data[4] ) [static]
```

8.1.3.14 optixGetCatmullRomVertexData() [2/2]

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

8.1.3.15 optixGetCatmullRomVertexDataFromHandle()

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

8.1.3.16 optixGetClusterId()

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

8.1.3.17 optixGetCubicBezierRocapsVertexData()

```
static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexData (
    float4 data[4] ) [static]
```

8.1.3.18 optixGetCubicBezierRocapsVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetCubicBezierRocapsVertexDataFromHandle (
    OptixTraversableHandle gas,
    unsigned int primIdx,
```

```

    unsigned int sbtGASIndex,
    float time,
    float4 data[4] ) [static]

```

8.1.3.19 optixGetCubicBezierVertexData() [1/2]

```

static __forceinline__ __device__ void optixGetCubicBezierVertexData (
    float4 data[4] ) [static]

```

8.1.3.20 optixGetCubicBezierVertexData() [2/2]

```

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

```

8.1.3.21 optixGetCubicBezierVertexDataFromHandle()

```

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

```

8.1.3.22 optixGetCubicBSplineRocapsVertexData()

```

static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexData
(
    float4 data[4] ) [static]

```

8.1.3.23 optixGetCubicBSplineRocapsVertexDataFromHandle()

```

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

```

8.1.3.24 optixGetCubicBSplineVertexData() [1/2]

```

static __forceinline__ __device__ void optixGetCubicBSplineVertexData (
    float4 data[4] ) [static]

```

8.1.3.25 optixGetCubicBSplineVertexData() [2/2]

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

8.1.3.26 optixGetCubicBSplineVertexDataFromHandle()

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

8.1.3.27 optixGetCurveParameter()

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

8.1.3.28 optixGetExceptionCode()

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

8.1.3.29 optixGetExceptionDetail_0()

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

8.1.3.30 optixGetExceptionDetail_1()

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

8.1.3.31 optixGetExceptionDetail_2()

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

8.1.3.32 optixGetExceptionDetail_3()

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

8.1.3.33 optixGetExceptionDetail_4()

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

8.1.3.34 optixGetExceptionDetail_5()

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

8.1.3.35 optixGetExceptionDetail_6()

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

8.1.3.36 optixGetExceptionDetail_7()

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

8.1.3.37 optixGetExceptionLineInfo()

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

8.1.3.38 optixGetGASMotionStepCount()

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

8.1.3.39 optixGetGASMotionTimeBegin()

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

8.1.3.40 optixGetGASMotionTimeEnd()

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

8.1.3.41 optixGetGASPointerFromHandle()

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

8.1.3.42 optixGetGASTraversableHandle()

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

8.1.3.43 optixGetHitKind()

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

8.1.3.44 optixGetInstanceChildFromHandle()

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

8.1.3.45 optixGetInstanceId()

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

8.1.3.46 optixGetInstanceIdFromHandle()

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

8.1.3.47 optixGetInstanceIndex()

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

8.1.3.48 optixGetInstanceInverseTransformFromHandle()

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

8.1.3.49 optixGetInstanceTransformFromHandle()

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

8.1.3.50 optixGetInstanceTraversableFromIAS()

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

8.1.3.51 optixGetLaunchDimensions()

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

8.1.3.52 optixGetLaunchIndex()

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

8.1.3.53 optixGetLinearCurveVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetLinearCurveVertexData (  
    float4 data[2] ) [static]
```

8.1.3.54 optixGetLinearCurveVertexData() [2/2]

```
static __forceinline__ __device__ void optixGetLinearCurveVertexData (  
    OptixTraversableHandle gas,  
    unsigned int primIdx,
```

```
    unsigned int sbtGASIndex,
    float time,
    float4 data[2] ) [static]
```

8.1.3.55 optixGetLinearCurveVertexDataFromHandle()

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

8.1.3.56 optixGetMatrixMotionTransformFromHandle()

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

8.1.3.57 optixGetObjectRayDirection()

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

8.1.3.58 optixGetObjectRayOrigin()

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

8.1.3.59 optixGetObjectToWorldTransformMatrix() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix
(
    const HitState & hs,
    float m[12] ) [static]
```

8.1.3.60 optixGetObjectToWorldTransformMatrix() [2/2]

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

8.1.3.61 optixGetPayload_0()

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

8.1.3.62 optixGetPayload_1()

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

8.1.3.63 optixGetPayload_10()

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

8.1.3.64 optixGetPayload_11()

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

8.1.3.65 optixGetPayload_12()

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

8.1.3.66 optixGetPayload_13()

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

8.1.3.67 optixGetPayload_14()

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

8.1.3.68 optixGetPayload_15()

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

8.1.3.69 optixGetPayload_16()

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

8.1.3.70 optixGetPayload_17()

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

8.1.3.71 optixGetPayload_18()

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

8.1.3.72 optixGetPayload_19()

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

8.1.3.73 optixGetPayload_2()

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

8.1.3.74 optixGetPayload_20()

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

8.1.3.75 optixGetPayload_21()

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

8.1.3.76 optixGetPayload_22()

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

8.1.3.77 optixGetPayload_23()

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

8.1.3.78 optixGetPayload_24()

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

8.1.3.79 optixGetPayload_25()

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

8.1.3.80 optixGetPayload_26()

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

8.1.3.81 optixGetPayload_27()

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

8.1.3.82 optixGetPayload_28()

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

8.1.3.83 optixGetPayload_29()

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

8.1.3.84 optixGetPayload_3()

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

8.1.3.85 optixGetPayload_30()

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

8.1.3.86 optixGetPayload_31()

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

8.1.3.87 optixGetPayload_4()

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

8.1.3.88 optixGetPayload_5()

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

8.1.3.89 optixGetPayload_6()

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

8.1.3.90 optixGetPayload_7()

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

8.1.3.91 optixGetPayload_8()

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

8.1.3.92 optixGetPayload_9()

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

8.1.3.93 optixGetPrimitiveIndex()

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

8.1.3.94 optixGetPrimitiveType() [1/2]

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

8.1.3.95 optixGetPrimitiveType() [2/2]

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

8.1.3.96 optixGetQuadraticBSplineRocapsVertexData()

```
static __forceinline__ __device__ void  
optixGetQuadraticBSplineRocapsVertexData ( )  
    float4 data[3] ) [static]
```

8.1.3.97 optixGetQuadraticBSplineRocapsVertexDataFromHandle()

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

8.1.3.98 optixGetQuadraticBSplineVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData ( )  
    float4 data[3] ) [static]
```

8.1.3.99 optixGetQuadraticBSplineVertexData() [2/2]

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

8.1.3.100 optixGetQuadraticBSplineVertexDataFromHandle()

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

8.1.3.101 optixGetRayFlags()

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

8.1.3.102 optixGetRayTime()

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

8.1.3.103 optixGetRayTmax()

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

8.1.3.104 optixGetRayTmin()

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

8.1.3.105 optixGetRayVisibilityMask()

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

8.1.3.106 optixGetRibbonNormal() [1/2]

```
static __forceinline__ __device__ float3 optixGetRibbonNormal (
    float2 ribbonParameters ) [static]
```

8.1.3.107 optixGetRibbonNormal() [2/2]

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

8.1.3.108 optixGetRibbonNormalFromHandle()

```
static __forceinline__ __device__ float3 optixGetRibbonNormalFromHandle (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
```

```
float time,
float2 ribbonParameters ) [static]
```

8.1.3.109 optixGetRibbonParameters()

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

8.1.3.110 optixGetRibbonVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetRibbonVertexData (
    float4 data[3] ) [static]
```

8.1.3.111 optixGetRibbonVertexData() [2/2]

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

8.1.3.112 optixGetRibbonVertexDataFromHandle()

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

8.1.3.113 optixGetSbtDataPointer()

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

8.1.3.114 optixGetSbtGASIndex()

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

8.1.3.115 optixGetSphereData() [1/2]

```
static __forceinline__ __device__ void optixGetSphereData (
    float4 data[1] ) [static]
```

8.1.3.116 optixGetSphereData() [2/2]

```
static __forceinline__ __device__ void optixGetSphereData (
    OptixTraversableHandle gas,
    unsigned int primIdx,
    unsigned int sbtGASIndex,
    float time,
```

```
float4 data[1] ) [static]
```

8.1.3.117 optixGetSphereDataFromHandle()

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

8.1.3.118 optixGetSRTMotionTransformFromHandle()

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

8.1.3.119 optixGetStaticTransformFromHandle()

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

8.1.3.120 optixGetTransformListHandle()

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

8.1.3.121 optixGetTransformListSize()

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

8.1.3.122 optixGetTransformTypeFromHandle()

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

8.1.3.123 optixGetTriangleBarycentrics()

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

8.1.3.124 optixGetTriangleVertexData() [1/2]

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

8.1.3.125 optixGetTriangleVertexData() [2/2]

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

8.1.3.126 optixGetTriangleVertexDataFromHandle()

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

8.1.3.127 optixGetWorldRayDirection()

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

8.1.3.128 optixGetWorldRayOrigin()

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

8.1.3.129 optixGetWorldToObjectTransformMatrix() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix
(
    const HitState & hs,
    float m[12] ) [static]
```

8.1.3.130 optixGetWorldToObjectTransformMatrix() [2/2]

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

8.1.3.131 optixHitObjectGetAttribute_0()

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

8.1.3.132 optixHitObjectGetAttribute_1()

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

8.1.3.133 optixHitObjectGetAttribute_2()

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

8.1.3.134 optixHitObjectGetAttribute_3()

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

8.1.3.135 optixHitObjectGetAttribute_4()

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

8.1.3.136 optixHitObjectGetAttribute_5()

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

8.1.3.137 optixHitObjectGetAttribute_6()

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

8.1.3.138 optixHitObjectGetAttribute_7()

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

8.1.3.139 optixHitObjectGetCatmullRomRocapsVertexData()

```
static __forceinline__ __device__ void  
optixHitObjectGetCatmullRomRocapsVertexData (   
    float4 data[4] ) [static]
```

8.1.3.140 optixHitObjectGetCatmullRomVertexData()

```
static __forceinline__ __device__ void optixHitObjectGetCatmullRomVertexData  
(   
    float4 data[4] ) [static]
```

8.1.3.141 optixHitObjectGetClusterId()

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

8.1.3.142 optixHitObjectGetCubicBezierRocapsVertexData()

```
static __forceinline__ __device__ void  
optixHitObjectGetCubicBezierRocapsVertexData (   
    float4 data[4] ) [static]
```

8.1.3.143 optixHitObjectGetCubicBezierVertexData()

```
static __forceinline__ __device__ void  
optixHitObjectGetCubicBezierVertexData (  
    float4 data[4] ) [static]
```

8.1.3.144 optixHitObjectGetCubicBSplineRocapsVertexData()

```
static __forceinline__ __device__ void  
optixHitObjectGetCubicBSplineRocapsVertexData (  
    float4 data[4] ) [static]
```

8.1.3.145 optixHitObjectGetCubicBSplineVertexData()

```
static __forceinline__ __device__ void  
optixHitObjectGetCubicBSplineVertexData (  
    float4 data[4] ) [static]
```

8.1.3.146 optixHitObjectGetCurveParameter()

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

8.1.3.147 optixHitObjectGetGASTraversableHandle()

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

8.1.3.148 optixHitObjectGetHitKind()

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

8.1.3.149 optixHitObjectGetInstanceId()

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

8.1.3.150 optixHitObjectGetInstanceIndex()

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

8.1.3.151 optixHitObjectGetLinearCurveVertexData()

```
static __forceinline__ __device__ void  
optixHitObjectGetLinearCurveVertexData (  
    float4 data[2] ) [static]
```

8.1.3.152 optixHitObjectGetObjectToWorldTransformMatrix()

```
static __forceinline__ __device__ void  
optixHitObjectGetObjectToWorldTransformMatrix (
```

```
float m[12] ) [static]
```

8.1.3.153 optixHitObjectGetPrimitiveIndex()

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

8.1.3.154 optixHitObjectGetQuadraticBSplineRocapsVertexData()

```
static __forceinline__ __device__ void  
optixHitObjectGetQuadraticBSplineRocapsVertexData (   
    float4 data[3] ) [static]
```

8.1.3.155 optixHitObjectGetQuadraticBSplineVertexData()

```
static __forceinline__ __device__ void  
optixHitObjectGetQuadraticBSplineVertexData (   
    float4 data[3] ) [static]
```

8.1.3.156 optixHitObjectGetRayFlags()

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

8.1.3.157 optixHitObjectGetRayTime()

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

8.1.3.158 optixHitObjectGetRayTmax()

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

8.1.3.159 optixHitObjectGetRayTmin()

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

8.1.3.160 optixHitObjectGetRibbonNormal()

```
static __forceinline__ __device__ float3 optixHitObjectGetRibbonNormal (   
    float2 ribbonParameters ) [static]
```

8.1.3.161 optixHitObjectGetRibbonParameters()

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

8.1.3.162 optixHitObjectGetRibbonVertexData()

```
static __forceinline__ __device__ void optixHitObjectGetRibbonVertexData (   
    float4 data[3] ) [static]
```

8.1.3.163 optixHitObjectGetSbtDataPointer()

```
static __forceinline__ __device__ CUdeviceptr
```

`optixHitObjectGetSbtDataPointer ( ) [static]`

8.1.3.164 `optixHitObjectGetSbtGASIndex()`

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

8.1.3.165 `optixHitObjectGetSbtRecordIndex()`

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

8.1.3.166 `optixHitObjectGetSphereData()`

`static __forceinline__ __device__ void optixHitObjectGetSphereData ( float4 data[1] ) [static]`

8.1.3.167 `optixHitObjectGetTransformListHandle()`

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

8.1.3.168 `optixHitObjectGetTransformListSize()`

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

8.1.3.169 `optixHitObjectGetTraverseData()`

`static __forceinline__ __device__ void optixHitObjectGetTraverseData ( OptixTraverseData * data ) [static]`

8.1.3.170 `optixHitObjectGetTriangleBarycentrics()`

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

8.1.3.171 `optixHitObjectGetTriangleVertexData()`

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

8.1.3.172 `optixHitObjectGetWorldRayDirection()`

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

8.1.3.173 `optixHitObjectGetWorldRayOrigin()`

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

8.1.3.174 optixHitObjectGetWorldToObjectTransformMatrix()

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

8.1.3.175 optixHitObjectIsHit()

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

8.1.3.176 optixHitObjectIsMiss()

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

8.1.3.177 optixHitObjectIsNop()

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

8.1.3.178 optixHitObjectSetSbtRecordIndex()

```
static __forceinline__ __device__ void optixHitObjectSetSbtRecordIndex (
    unsigned int sbtRecordIndex ) [static]
```

8.1.3.179 optixHitObjectTransformNormalFromObjectToWorldSpace()

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

8.1.3.180 optixHitObjectTransformNormalFromWorldToObjectSpace()

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

8.1.3.181 optixHitObjectTransformPointFromObjectToWorldSpace()

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

8.1.3.182 optixHitObjectTransformPointFromWorldToObjectSpace()

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

8.1.3.183 optixHitObjectTransformVectorFromObjectToWorldSpace()

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

8.1.3.184 optixHitObjectTransformVectorFromWorldToObjectSpace()

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

8.1.3.185 optixIgnoreIntersection()

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

8.1.3.186 optixInvoke() [1/2]

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

8.1.3.187 optixInvoke() [2/2]

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

8.1.3.188 optixIsBackFaceHit() [1/2]

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

8.1.3.189 optixIsBackFaceHit() [2/2]

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

8.1.3.190 optixIsFrontFaceHit() [1/2]

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

8.1.3.191 optixIsFrontFaceHit() [2/2]

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

8.1.3.192 optixIsTriangleBackFaceHit()

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

8.1.3.193 optixIsTriangleFrontFaceHit()

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

8.1.3.194 optixIsTriangleHit()

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

8.1.3.195 optixMakeHitObject()

```
static __forceinline__ __device__ void optixMakeHitObject (
    OptixTraversableHandle handle,
    float3 rayOrigin,
    float3 rayDirection,
    float tmin,
    float rayTime,
    unsigned int rayFlags,
    OptixTraverseData traverseData,
    const OptixTraversableHandle * transforms,
    unsigned int numTransforms ) [static]
```

8.1.3.196 optixMakeMissHitObject()

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

8.1.3.197 optixMakeNopHitObject()

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

8.1.3.198 optixReorder() [1/2]

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

8.1.3.199 optixReorder() [2/2]

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

8.1.3.200 optixReportIntersection() [1/9]

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

8.1.3.201 optixReportIntersection() [2/9]

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

```
    unsigned int a0 ) [static]
```

8.1.3.202 optixReportIntersection() [3/9]

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

8.1.3.203 optixReportIntersection() [4/9]

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

8.1.3.204 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]
```

8.1.3.205 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]
```

8.1.3.206 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]

```

8.1.3.207 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]

```

8.1.3.208 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]

```

8.1.3.209 optixSetPayload_0()

```

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

```

8.1.3.210 optixSetPayload_1()

```

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

```

8.1.3.211 optixSetPayload_10()

```

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

```

8.1.3.212 optixSetPayload_11()

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

8.1.3.213 optixSetPayload_12()

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

8.1.3.214 optixSetPayload_13()

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

8.1.3.215 optixSetPayload_14()

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

8.1.3.216 optixSetPayload_15()

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

8.1.3.217 optixSetPayload_16()

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

8.1.3.218 optixSetPayload_17()

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

8.1.3.219 optixSetPayload_18()

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

8.1.3.220 optixSetPayload_19()

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

8.1.3.221 optixSetPayload_20()

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

8.1.3.222 optixSetPayload_20()

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

unsigned int *p*) *[static]*

8.1.3.223 optixSetPayload_21()

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

8.1.3.224 optixSetPayload_22()

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

8.1.3.225 optixSetPayload_23()

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

8.1.3.226 optixSetPayload_24()

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

8.1.3.227 optixSetPayload_25()

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

8.1.3.228 optixSetPayload_26()

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

8.1.3.229 optixSetPayload_27()

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

8.1.3.230 optixSetPayload_28()

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

8.1.3.231 optixSetPayload_29()

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

8.1.3.232 optixSetPayload_3()

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

8.1.3.233 optixSetPayload_30()

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

8.1.3.234 optixSetPayload_31()

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

8.1.3.235 optixSetPayload_4()

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

8.1.3.236 optixSetPayload_5()

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

8.1.3.237 optixSetPayload_6()

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

8.1.3.238 optixSetPayload_7()

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

8.1.3.239 optixSetPayload_8()

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

8.1.3.240 optixSetPayload_9()

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

8.1.3.241 optixSetPayloadTypes()

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

8.1.3.242 optixTerminateRay()

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

8.1.3.243 optixTexFootprint2D()

```
static __forceinline__ __device__ uint4 optixTexFootprint2D (  
    unsigned long long tex,
```

```

    unsigned int texInfo,
    float x,
    float y,
    unsigned int * singleMipLevel ) [static]

```

8.1.3.244 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]

```

8.1.3.245 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]

```

8.1.3.246 optixThrowException() [1/9]

```

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

```

8.1.3.247 optixThrowException() [2/9]

```

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

```

8.1.3.248 optixThrowException() [3/9]

```

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

```

8.1.3.249 optixThrowException() [4/9]

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

8.1.3.250 optixThrowException() [5/9]

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

8.1.3.251 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]
```

8.1.3.252 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]
```

8.1.3.253 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]
```

8.1.3.254 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]
```

8.1.3.255 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]
```

8.1.3.256 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]

```

8.1.3.257 optixTransformNormalFromObjectToWorldSpace() [1/2]

```

template<typename HitState >
static __forceinline__ __device__ float3
optixTransformNormalFromObjectToWorldSpace (
    const HitState & hs,
    float3 normal ) [static]

```

8.1.3.258 optixTransformNormalFromObjectToWorldSpace() [2/2]

```

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

```

8.1.3.259 optixTransformNormalFromWorldToObjectSpace() [1/2]

```

template<typename HitState >
static __forceinline__ __device__ float3
optixTransformNormalFromWorldToObjectSpace (
    const HitState & hs,
    float3 normal ) [static]

```

8.1.3.260 optixTransformNormalFromWorldToObjectSpace() [2/2]

```

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

```

8.1.3.261 optixTransformPointFromObjectToWorldSpace() [1/2]

```

template<typename HitState >
static __forceinline__ __device__ float3
optixTransformPointFromObjectToWorldSpace (
    const HitState & hs,
    float3 point ) [static]

```

8.1.3.262 optixTransformPointFromObjectToWorldSpace() [2/2]

```

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

```

8.1.3.263 optixTransformPointFromWorldToObjectSpace() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformPointFromWorldToObjectSpace (
    const HitState & hs,
    float3 point ) [static]
```

8.1.3.264 optixTransformPointFromWorldToObjectSpace() [2/2]

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

8.1.3.265 optixTransformVectorFromObjectToWorldSpace() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformVectorFromObjectToWorldSpace (
    const HitState & hs,
    float3 vec ) [static]
```

8.1.3.266 optixTransformVectorFromObjectToWorldSpace() [2/2]

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

8.1.3.267 optixTransformVectorFromWorldToObjectSpace() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformVectorFromWorldToObjectSpace (
    const HitState & hs,
    float3 vec ) [static]
```

8.1.3.268 optixTransformVectorFromWorldToObjectSpace() [2/2]

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

8.1.3.269 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]

```

8.1.3.270 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]

```

8.1.3.271 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
13 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
14 #error("optix_device_impl.h is an internal header file and must not be used directly. Please use
15 optix_device.h or optix.h instead.")
16 #endif
17
18 #ifndef OPTIX_DEVICE_IMPL_H

```

```

25 #define OPTIX_DEVICE_IMPL_H
26
27 #include "internal/optix_device_impl_transformations.h"
28
29 #ifndef __CUDACC_RTC__
30 #include <initializer_list>
31 #include <type_traits>
32 #endif
33
34 namespace optix_internal {
35 template <typename...>
36 struct TypePack{};
37 } // namespace optix_internal
38
39 template <typename... Payload>
40 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
41                                                    float3 rayOrigin,
42                                                    float3 rayDirection,
43                                                    float tmin,
44                                                    float tmax,
45                                                    float rayTime,
46                                                    OptixVisibilityMask visibilityMask,
47                                                    unsigned int rayFlags,
48                                                    unsigned int SBTOffset,
49                                                    unsigned int SBTstride,
50                                                    unsigned int missSBTIndex,
51                                                    Payload&... payload)
52 {
53     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
54     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
55     // TypePack 1    unsigned int    T0    T1    T2    ...    Tn-1    Tn
56     // TypePack 2      T0            T1    T2    T3    ...    Tn      unsigned int
57 #ifndef __CUDACC_RTC__
58     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
59 optix_internal::TypePack<Payload..., unsigned int>::value,
60 "All payload parameters need to be unsigned int.");
61 #endif
62     OptixPayloadTypeID type = OPTIX_PAYLOAD_TYPE_DEFAULT;
63     float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
64     float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
65     unsigned int p[33] = { 0, payload... };
66     int payloadSize = (int)sizeof...(Payload);
67     asm volatile(
68         "call"
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,%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,%59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
74         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
75         "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
76         "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
77         "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
78         "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
79         : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
80         "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBTOffset), "r"(SBTstride),
81         "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
82         "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
83         "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
84         "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
85         "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
86         :);
87     unsigned int index = 1;
88     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};

```

```

89 }
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 __CUDACC_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         "(%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),"
122         "_optix_hitobject_traverse,"
123         "(%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);"
124         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
125         "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
126         "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
127         "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
128         "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
129         : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
130         "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBTOffset), "r"(SBTstride),
131         "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
132         "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
133         "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
134         "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
135         "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
136         :);
137     unsigned int index = 1;
138     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
139 }
140
141 template <typename... Payload>
142 static __forceinline__ __device__ void optixTrace(OptixPayloadTypeID type,
143 OptixTraversableHandle handle,
144 float3 rayOrigin,
145 float3 rayDirection,
146 float tmin,
147 float tmax,
148 float rayTime,
149 OptixVisibilityMask visibilityMask,
150 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 __CUDACC_RTC__
163     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
164 optix_internal::TypePack<Payload..., unsigned int>::value,
165     "All payload parameters need to be unsigned int.");
166     #endif
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         "(%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);"
178         : "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]), "r"(p[6]), "r"(p[7]),
179         "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]), "r"(p[14]),
180         "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]), "r"(p[21]),
181         "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]), "r"(p[28]),
182         "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
183         : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
184         "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBTOffset), "r"(SBTstride),
185         "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
186         "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
187         "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
188         "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
189         "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
190         :);
191     unsigned int index = 1;
192     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
193 }
194
195 template <typename... Payload>
196 static __forceinline__ __device__ void optixTraverse(OptixPayloadTypeID      type,
197 OptixTraversableHandle handle,
198 float3      rayOrigin,
199 float3      rayDirection,
200 float      tmin,
201 float      tmax,
202 float      rayTime,
203 OptixVisibilityMask visibilityMask,
204 unsigned int rayFlags,
205 unsigned int SBTOffset,
206 unsigned int SBTstride,
207 unsigned int missSBTIndex,
208 Payload&... payload)
209 {
210     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
211     // TypePack 1      unsigned int      T0      T1      T2      ...      Tn-1      Tn
212     // TypePack 2      T0      T1      T2      T3      ...      Tn      unsigned int
213     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
214     #ifndef __CUDACC_RTC__
215     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,

```

```

optix_internal::TypePack<Payload..., unsigned int>::value,
217         "All payload parameters need to be unsigned int.");
218 #endif
219
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
numCoherenceHintBits)
249 {
250     asm volatile(
251         "call"
252         "(),"
253         "_optix_hitobject_reorder,"
254         "(%0,%1);"
255         :
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
292         "(%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), "
293         "_optix_hitobject_invoke, "
294
295         "(%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); "
296         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
297         "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
298         "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
299         "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
300         "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
301         : "r"(type), "r"(payloadSize), "r"(p[1]), "r"(p[2]),
302         "r"(p[3]), "r"(p[4]), "r"(p[5]), "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]),
303         "r"(p[11]), "r"(p[12]), "r"(p[13]), "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]),
304         "r"(p[18]), "r"(p[19]), "r"(p[20]), "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]),
305         "r"(p[25]), "r"(p[26]), "r"(p[27]), "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
306         :);
307
308     unsigned int index = 1;
309     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
310 }
311
312 template <typename... Payload>
313 static __forceinline__ __device__ void optixInvoke(Payload&... payload)
314 {
315     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
316     // TypePack 1      unsigned int    T0      T1      T2      ...   Tn-1      Tn
317     // TypePack 2      T0              T1      T2      T3      ...   Tn          unsigned int
318     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
319 #ifndef __CUDACC_RTC__
320     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int>::value,
321         "All payload parameters need to be unsigned int.");
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 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle handle,
352                                                         float3 rayOrigin,
353                                                         float3 rayDirection,
354                                                         float tmin,
355                                                         float rayTime,
356                                                         unsigned int rayFlags,
357                                                         OptixTraverseData traverseData,
358                                                         const OptixTraversableHandle* transforms,
359                                                         unsigned int numTransforms)
360 {
361     float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
362     float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
363
364     asm volatile(
365         "call\n"
366         "(),\n"
367         "_optix_hitobject_make_with_traverse_data_v2,\n"
368         "(%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,%32),\n"
369         ":\n"
370         ": \"l\"(handle), \"f\"(ox), \"f\"(oy), \"f\"(oz), \"f\"(dx), \"f\"(dy), \"f\"(dz), \"f\"(tmin), \"f\"(rayTime),\n"
371         "\"r\"(rayFlags),\n"
372         "\"r\"(traverseData.data[0]), \"r\"(traverseData.data[1]), \"r\"(traverseData.data[2]),\n"
373         "\"r\"(traverseData.data[3]), \"r\"(traverseData.data[4]), \"r\"(traverseData.data[5]),\n"
374         "\"r\"(traverseData.data[6]), \"r\"(traverseData.data[7]), \"r\"(traverseData.data[8]),\n"
375         "\"r\"(traverseData.data[9]), \"r\"(traverseData.data[10]), \"r\"(traverseData.data[11]),\n"
376         "\"r\"(traverseData.data[12]), \"r\"(traverseData.data[13]), \"r\"(traverseData.data[14]),\n"
377         "\"r\"(traverseData.data[15]), \"r\"(traverseData.data[16]), \"r\"(traverseData.data[17]),\n"
378         "\"r\"(traverseData.data[18]), \"r\"(traverseData.data[19]), \"l\"(transforms), \"r\"(numTransforms)\n"
379         ":\n"
380     );
381
382     static __forceinline__ __device__ void optixMakeMissHitObject(unsigned int missSBTIndex,
383                                                                    float3 rayOrigin,
384                                                                    float3 rayDirection,
385                                                                    float tmin,
386                                                                    float tmax,
387                                                                    float rayTime,
388                                                                    unsigned int rayFlags)
389     {
390         float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
391         float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
392
393         asm volatile(
394             "call\n"
395             "(),\n"
396             "_optix_hitobject_make_miss_v2,\n"
397             "(%0,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10),\n"
398             ":\n"
399             ": \"r\"(missSBTIndex), \"f\"(ox), \"f\"(oy), \"f\"(oz), \"f\"(dx), \"f\"(dy), \"f\"(dz), \"f\"(tmin),\n"
400             "\"f\"(tmax), \"f\"(rayTime), \"r\"(rayFlags)\n"
401             ":\n"
402         );
403     }
404
405     static __forceinline__ __device__ void optixMakeNopHitObject()
406     {
407         asm volatile(

```

```

406         "call"
407         "(", "
408         "_optix_hitobject_make_nop,"
409         "());"
410         :
411         :
412         :);
413     }
414
415     static __forceinline__ __device__ void optixHitObjectGetTraverseData(OptixTraverseData* data)
416     {
417         asm volatile(
418             "call"
419             "(%0,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18,%19), "
420             "_optix_hitobject_get_traverse_data,"
421             "());"
422             : "=r"(data->data[0]), "=r"(data->data[1]), "=r"(data->data[2]), "=r"(data->data[3]),
423             "=r"(data->data[4]),
424             "=r"(data->data[5]), "=r"(data->data[6]), "=r"(data->data[7]), "=r"(data->data[8]),
425             "=r"(data->data[9]),
426             "=r"(data->data[10]), "=r"(data->data[11]), "=r"(data->data[12]), "=r"(data->data[13]),
427             "=r"(data->data[14]),
428             "=r"(data->data[15]), "=r"(data->data[16]), "=r"(data->data[17]), "=r"(data->data[18]),
429             "=r"(data->data[19])
430             :
431             :);
432     }
433
434     static __forceinline__ __device__ bool optixHitObjectIsHit()
435     {
436         unsigned int result;
437         asm volatile(
438             "call (%0), _optix_hitobject_is_hit,"
439             "());"
440             : "=r"(result)
441             :
442             :);
443         return result;
444     }
445
446     static __forceinline__ __device__ bool optixHitObjectIsMiss()
447     {
448         unsigned int result;
449         asm volatile(
450             "call (%0), _optix_hitobject_is_miss,"
451             "());"
452             : "=r"(result)
453             :
454             :);
455         return result;
456     }
457
458     static __forceinline__ __device__ bool optixHitObjectIsNop()
459     {
460         unsigned int result;
461         asm volatile(
462             "call (%0), _optix_hitobject_is_nop,"
463             "());"
464             : "=r"(result)
465             :
466             :);
467         return result;
468     }
469
470     static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId()
471     {
472         unsigned int result;

```

```

469     asm volatile(
470         "call (%0), _optix_hitobject_get_instance_id, "
471         "();"
472         : "=r"(result)
473         :
474         :);
475     return result;
476 }
477
478 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex()
479 {
480     unsigned int result;
481     asm volatile(
482         "call (%0), _optix_hitobject_get_instance_idx, "
483         "();"
484         : "=r"(result)
485         :
486         :);
487     return result;
488 }
489
490 static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex()
491 {
492     unsigned int result;
493     asm volatile(
494         "call (%0), _optix_hitobject_get_primitive_idx, "
495         "();"
496         : "=r"(result)
497         :
498         :);
499     return result;
500 }
501
502 static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize()
503 {
504     unsigned int result;
505     asm volatile(
506         "call (%0), _optix_hitobject_get_transform_list_size, "
507         "();"
508         : "=r"(result)
509         :
510         :);
511     return result;
512 }
513
514 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle(unsigned
int index)
515 {
516     unsigned long long result;
517     asm volatile(
518         "call (%0), _optix_hitobject_get_transform_list_handle, "
519         "(%1);"
520         : "=l"(result)
521         : "r"(index)
522         :);
523     return result;
524 }
525
526 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex()
527 {
528     unsigned int result;
529     asm volatile(
530         "call (%0), _optix_hitobject_get_sbt_gas_idx, "
531         "();"
532         : "=r"(result)
533         :
534         :);

```

```

535     return result;
536 }
537
538 static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind()
539 {
540     unsigned int result;
541     asm volatile(
542         "call (%0), _optix_hitobject_get_hitkind,"
543         "();"
544         : "=r"(result)
545         :
546         :);
547     return result;
548 }
549
550 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin()
551 {
552     float x, y, z;
553     asm volatile(
554         "call (%0), _optix_hitobject_get_world_ray_origin_x,"
555         "();"
556         : "=f"(x)
557         :
558         :);
559     asm volatile(
560         "call (%0), _optix_hitobject_get_world_ray_origin_y,"
561         "();"
562         : "=f"(y)
563         :
564         :);
565     asm volatile(
566         "call (%0), _optix_hitobject_get_world_ray_origin_z,"
567         "();"
568         : "=f"(z)
569         :
570         :);
571     return make_float3(x, y, z);
572 }
573
574 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection()
575 {
576     float x, y, z;
577     asm volatile(
578         "call (%0), _optix_hitobject_get_world_ray_direction_x,"
579         "();"
580         : "=f"(x)
581         :
582         :);
583     asm volatile(
584         "call (%0), _optix_hitobject_get_world_ray_direction_y,"
585         "();"
586         : "=f"(y)
587         :
588         :);
589     asm volatile(
590         "call (%0), _optix_hitobject_get_world_ray_direction_z,"
591         "();"
592         : "=f"(z)
593         :
594         :);
595     return make_float3(x, y, z);
596 }
597
598 static __forceinline__ __device__ float optixHitObjectGetRayTmin()
599 {
600     float result;
601     asm volatile(

```

```

602         "call (%0), _optix_hitobject_get_ray_tmin,"
603         "();"
604         : "=f"(result)
605         :
606         :);
607     return result;
608 }
609
610 static __forceinline__ __device__ float optixHitObjectGetRayTmax()
611 {
612     float result;
613     asm volatile(
614         "call (%0), _optix_hitobject_get_ray_tmax,"
615         "();"
616         : "=f"(result)
617         :
618         :);
619     return result;
620 }
621
622 static __forceinline__ __device__ float optixHitObjectGetRayTime()
623 {
624     float result;
625     asm volatile(
626         "call (%0), _optix_hitobject_get_ray_time,"
627         "();"
628         : "=f"(result)
629         :
630         :);
631     return result;
632 }
633
634 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0()
635 {
636     unsigned int ret;
637     asm volatile(
638         "call (%0), _optix_hitobject_get_attribute,"
639         "(%1);"
640         : "=r"(ret)
641         : "r"(0)
642         :);
643     return ret;
644 }
645
646 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1()
647 {
648     unsigned int ret;
649     asm volatile(
650         "call (%0), _optix_hitobject_get_attribute,"
651         "(%1);"
652         : "=r"(ret)
653         : "r"(1)
654         :);
655     return ret;
656 }
657
658 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2()
659 {
660     unsigned int ret;
661     asm volatile(
662         "call (%0), _optix_hitobject_get_attribute,"
663         "(%1);"
664         : "=r"(ret)
665         : "r"(2)
666         :);
667     return ret;
668 }

```

```

669
670 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3()
671 {
672     unsigned int ret;
673     asm volatile(
674         "call (%0), _optix_hitobject_get_attribute,"
675         "(%1);"
676         : "=r"(ret)
677         : "r"(3)
678         :);
679     return ret;
680 }
681
682 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4()
683 {
684     unsigned int ret;
685     asm volatile(
686         "call (%0), _optix_hitobject_get_attribute,"
687         "(%1);"
688         : "=r"(ret)
689         : "r"(4)
690         :);
691     return ret;
692 }
693
694 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5()
695 {
696     unsigned int ret;
697     asm volatile(
698         "call (%0), _optix_hitobject_get_attribute,"
699         "(%1);"
700         : "=r"(ret)
701         : "r"(5)
702         :);
703     return ret;
704 }
705
706 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6()
707 {
708     unsigned int ret;
709     asm volatile(
710         "call (%0), _optix_hitobject_get_attribute,"
711         "(%1);"
712         : "=r"(ret)
713         : "r"(6)
714         :);
715     return ret;
716 }
717
718 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7()
719 {
720     unsigned int ret;
721     asm volatile(
722         "call (%0), _optix_hitobject_get_attribute,"
723         "(%1);"
724         : "=r"(ret)
725         : "r"(7)
726         :);
727     return ret;
728 }
729
730 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex()
731 {
732     unsigned int result;
733     asm volatile(
734         "call (%0), _optix_hitobject_get_sbt_record_index,"
735         "();"

```

```

736         : "=r"(result)
737         :
738         :);
739     return result;
740 }
741
742 static __forceinline__ __device__ void optixHitObjectSetSbtRecordIndex(unsigned int sbtRecordIndex)
743 {
744     asm volatile(
745         "call (), _optix_hitobject_set_sbt_record_index,"
746         "(%0);"
747         :
748         : "r"(sbtRecordIndex)
749         :);
750 }
751
752 static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer()
753 {
754     unsigned long long ptr;
755     asm volatile(
756         "call (%0), _optix_hitobject_get_sbt_data_pointer,"
757         "();"
758         : "=l"(ptr)
759         :
760         :);
761     return ptr;
762 }
763
764
765 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetGASTraversableHandle()
766 {
767     unsigned long long handle;
768     asm("call (%0), _optix_hitobject_get_gas_traversable_handle, ();" : "=l"(handle) :);
769     return (OptixTraversableHandle)handle;
770 }
771
772
773 static __forceinline__ __device__ unsigned int optixHitObjectGetRayFlags()
774 {
775     unsigned int u0;
776     asm("call (%0), _optix_hitobject_get_ray_flags, ();" : "=r"(u0) :);
777     return u0;
778 }
779
780
781 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p)
782 {
783     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(0), "r"(p) :);
784 }
785
786 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p)
787 {
788     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(1), "r"(p) :);
789 }
790
791 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p)
792 {
793     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(2), "r"(p) :);
794 }
795
796 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p)
797 {
798     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(3), "r"(p) :);
799 }
800
801 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p)
802 {

```

```

803     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(4), "r"(p) :);
804 }
805
806 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p)
807 {
808     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(5), "r"(p) :);
809 }
810
811 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p)
812 {
813     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(6), "r"(p) :);
814 }
815
816 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p)
817 {
818     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(7), "r"(p) :);
819 }
820
821 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p)
822 {
823     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(8), "r"(p) :);
824 }
825
826 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p)
827 {
828     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(9), "r"(p) :);
829 }
830
831 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p)
832 {
833     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(10), "r"(p) :);
834 }
835
836 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p)
837 {
838     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(11), "r"(p) :);
839 }
840
841 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p)
842 {
843     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(12), "r"(p) :);
844 }
845
846 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p)
847 {
848     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(13), "r"(p) :);
849 }
850
851 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p)
852 {
853     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(14), "r"(p) :);
854 }
855
856 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p)
857 {
858     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(15), "r"(p) :);
859 }
860
861 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p)
862 {
863     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(16), "r"(p) :);
864 }
865
866 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p)
867 {
868     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(17), "r"(p) :);
869 }

```

```

870
871 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p)
872 {
873     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(18), "r"(p) :);
874 }
875
876 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p)
877 {
878     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(19), "r"(p) :);
879 }
880
881 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p)
882 {
883     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(20), "r"(p) :);
884 }
885
886 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p)
887 {
888     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(21), "r"(p) :);
889 }
890
891 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p)
892 {
893     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(22), "r"(p) :);
894 }
895
896 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p)
897 {
898     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(23), "r"(p) :);
899 }
900
901 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p)
902 {
903     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(24), "r"(p) :);
904 }
905
906 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p)
907 {
908     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(25), "r"(p) :);
909 }
910
911 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p)
912 {
913     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(26), "r"(p) :);
914 }
915
916 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p)
917 {
918     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(27), "r"(p) :);
919 }
920
921 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p)
922 {
923     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(28), "r"(p) :);
924 }
925
926 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p)
927 {
928     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(29), "r"(p) :);
929 }
930
931 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p)
932 {
933     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(30), "r"(p) :);
934 }
935
936 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p)

```

```

937 {
938     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(31), "r"(p) :);
939 }
940
941 static __forceinline__ __device__ unsigned int optixGetPayload_0()
942 {
943     unsigned int result;
944     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(0) :);
945     return result;
946 }
947
948 static __forceinline__ __device__ unsigned int optixGetPayload_1()
949 {
950     unsigned int result;
951     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(1) :);
952     return result;
953 }
954
955 static __forceinline__ __device__ unsigned int optixGetPayload_2()
956 {
957     unsigned int result;
958     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(2) :);
959     return result;
960 }
961
962 static __forceinline__ __device__ unsigned int optixGetPayload_3()
963 {
964     unsigned int result;
965     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(3) :);
966     return result;
967 }
968
969 static __forceinline__ __device__ unsigned int optixGetPayload_4()
970 {
971     unsigned int result;
972     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(4) :);
973     return result;
974 }
975
976 static __forceinline__ __device__ unsigned int optixGetPayload_5()
977 {
978     unsigned int result;
979     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(5) :);
980     return result;
981 }
982
983 static __forceinline__ __device__ unsigned int optixGetPayload_6()
984 {
985     unsigned int result;
986     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(6) :);
987     return result;
988 }
989
990 static __forceinline__ __device__ unsigned int optixGetPayload_7()
991 {
992     unsigned int result;
993     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(7) :);
994     return result;
995 }
996
997 static __forceinline__ __device__ unsigned int optixGetPayload_8()
998 {
999     unsigned int result;
1000     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(8) :);
1001     return result;
1002 }
1003

```

```

1004 static __forceinline__ __device__ unsigned int optixGetPayload_9()
1005 {
1006     unsigned int result;
1007     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(9) :);
1008     return result;
1009 }
1010
1011 static __forceinline__ __device__ unsigned int optixGetPayload_10()
1012 {
1013     unsigned int result;
1014     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(10) :);
1015     return result;
1016 }
1017
1018 static __forceinline__ __device__ unsigned int optixGetPayload_11()
1019 {
1020     unsigned int result;
1021     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(11) :);
1022     return result;
1023 }
1024
1025 static __forceinline__ __device__ unsigned int optixGetPayload_12()
1026 {
1027     unsigned int result;
1028     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(12) :);
1029     return result;
1030 }
1031
1032 static __forceinline__ __device__ unsigned int optixGetPayload_13()
1033 {
1034     unsigned int result;
1035     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(13) :);
1036     return result;
1037 }
1038
1039 static __forceinline__ __device__ unsigned int optixGetPayload_14()
1040 {
1041     unsigned int result;
1042     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(14) :);
1043     return result;
1044 }
1045
1046 static __forceinline__ __device__ unsigned int optixGetPayload_15()
1047 {
1048     unsigned int result;
1049     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(15) :);
1050     return result;
1051 }
1052
1053 static __forceinline__ __device__ unsigned int optixGetPayload_16()
1054 {
1055     unsigned int result;
1056     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(16) :);
1057     return result;
1058 }
1059
1060 static __forceinline__ __device__ unsigned int optixGetPayload_17()
1061 {
1062     unsigned int result;
1063     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(17) :);
1064     return result;
1065 }
1066
1067 static __forceinline__ __device__ unsigned int optixGetPayload_18()
1068 {
1069     unsigned int result;
1070     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(18) :);

```

```

1071     return result;
1072 }
1073
1074 static __forceinline__ __device__ unsigned int optixGetPayload_19()
1075 {
1076     unsigned int result;
1077     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(19) :);
1078     return result;
1079 }
1080
1081 static __forceinline__ __device__ unsigned int optixGetPayload_20()
1082 {
1083     unsigned int result;
1084     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(20) :);
1085     return result;
1086 }
1087
1088 static __forceinline__ __device__ unsigned int optixGetPayload_21()
1089 {
1090     unsigned int result;
1091     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(21) :);
1092     return result;
1093 }
1094
1095 static __forceinline__ __device__ unsigned int optixGetPayload_22()
1096 {
1097     unsigned int result;
1098     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(22) :);
1099     return result;
1100 }
1101
1102 static __forceinline__ __device__ unsigned int optixGetPayload_23()
1103 {
1104     unsigned int result;
1105     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(23) :);
1106     return result;
1107 }
1108
1109 static __forceinline__ __device__ unsigned int optixGetPayload_24()
1110 {
1111     unsigned int result;
1112     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(24) :);
1113     return result;
1114 }
1115
1116 static __forceinline__ __device__ unsigned int optixGetPayload_25()
1117 {
1118     unsigned int result;
1119     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(25) :);
1120     return result;
1121 }
1122
1123 static __forceinline__ __device__ unsigned int optixGetPayload_26()
1124 {
1125     unsigned int result;
1126     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(26) :);
1127     return result;
1128 }
1129
1130 static __forceinline__ __device__ unsigned int optixGetPayload_27()
1131 {
1132     unsigned int result;
1133     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(27) :);
1134     return result;
1135 }
1136
1137 static __forceinline__ __device__ unsigned int optixGetPayload_28()

```

```

1138 {
1139     unsigned int result;
1140     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(28) :);
1141     return result;
1142 }
1143
1144 static __forceinline__ __device__ unsigned int optixGetPayload_29()
1145 {
1146     unsigned int result;
1147     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(29) :);
1148     return result;
1149 }
1150
1151 static __forceinline__ __device__ unsigned int optixGetPayload_30()
1152 {
1153     unsigned int result;
1154     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(30) :);
1155     return result;
1156 }
1157
1158 static __forceinline__ __device__ unsigned int optixGetPayload_31()
1159 {
1160     unsigned int result;
1161     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(31) :);
1162     return result;
1163 }
1164
1165 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int types)
1166 {
1167     asm volatile("call _optix_set_payload_types, (%0);" : : "r"(types) :);
1168 }
1169
1170 static __forceinline__ __device__ unsigned int optixUndefinedValue()
1171 {
1172     unsigned int u0;
1173     asm("call (%0), _optix_undef_value, ();" : "=r"(u0) :);
1174     return u0;
1175 }
1176
1177 __device__ __forceinline__ unsigned int optixGetRemainingTraceDepth()
1178 {
1179     unsigned int result;
1180     asm (
1181         "call (%0), _optix_get_remaining_trace_depth,"
1182         "();"
1183         : "=r"(result)
1184         :
1185         :);
1186     return result;
1187 }
1188
1189 static __forceinline__ __device__ float3 optixGetWorldRayOrigin()
1190 {
1191     float f0, f1, f2;
1192     asm("call (%0), _optix_get_world_ray_origin_x, ();" : "=f"(f0) :);
1193     asm("call (%0), _optix_get_world_ray_origin_y, ();" : "=f"(f1) :);
1194     asm("call (%0), _optix_get_world_ray_origin_z, ();" : "=f"(f2) :);
1195     return make_float3(f0, f1, f2);
1196 }
1197
1198 static __forceinline__ __device__ float3 optixGetWorldRayDirection()
1199 {
1200     float f0, f1, f2;
1201     asm("call (%0), _optix_get_world_ray_direction_x, ();" : "=f"(f0) :);
1202     asm("call (%0), _optix_get_world_ray_direction_y, ();" : "=f"(f1) :);
1203     asm("call (%0), _optix_get_world_ray_direction_z, ();" : "=f"(f2) :);
1204     return make_float3(f0, f1, f2);

```

```

1205 }
1206
1207 static __forceinline__ __device__ float3 optixGetObjectRayOrigin()
1208 {
1209     float f0, f1, f2;
1210     asm("call (%0), _optix_get_object_ray_origin_x, ();" : "=f"(f0) :);
1211     asm("call (%0), _optix_get_object_ray_origin_y, ();" : "=f"(f1) :);
1212     asm("call (%0), _optix_get_object_ray_origin_z, ();" : "=f"(f2) :);
1213     return make_float3(f0, f1, f2);
1214 }
1215
1216 static __forceinline__ __device__ float3 optixGetObjectRayDirection()
1217 {
1218     float f0, f1, f2;
1219     asm("call (%0), _optix_get_object_ray_direction_x, ();" : "=f"(f0) :);
1220     asm("call (%0), _optix_get_object_ray_direction_y, ();" : "=f"(f1) :);
1221     asm("call (%0), _optix_get_object_ray_direction_z, ();" : "=f"(f2) :);
1222     return make_float3(f0, f1, f2);
1223 }
1224
1225 static __forceinline__ __device__ float optixGetRayTmin()
1226 {
1227     float f0;
1228     asm("call (%0), _optix_get_ray_tmin, ();" : "=f"(f0) :);
1229     return f0;
1230 }
1231
1232 static __forceinline__ __device__ float optixGetRayTmax()
1233 {
1234     float f0;
1235     asm("call (%0), _optix_get_ray_tmax, ();" : "=f"(f0) :);
1236     return f0;
1237 }
1238
1239 static __forceinline__ __device__ float optixGetRayTime()
1240 {
1241     float f0;
1242     asm("call (%0), _optix_get_ray_time, ();" : "=f"(f0) :);
1243     return f0;
1244 }
1245
1246 static __forceinline__ __device__ unsigned int optixGetRayFlags()
1247 {
1248     unsigned int u0;
1249     asm("call (%0), _optix_get_ray_flags, ();" : "=r"(u0) :);
1250     return u0;
1251 }
1252
1253 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask()
1254 {
1255     unsigned int u0;
1256     asm("call (%0), _optix_get_ray_visibility_mask, ();" : "=r"(u0) :);
1257     return u0;
1258 }
1259
1260 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias,
1261                                     instIdx)
1262 {
1263     unsigned long long handle;
1264     asm("call (%0), _optix_get_instance_traversable_from_ias, (%1, %2);"
        : "=l"(handle) : "l"(ias), "r"(instIdx));
1265     return (OptixTraversableHandle)handle;
1266 }
1267
1268
1269

```

unsigned int

```

1270 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas,
1271                                                                    unsigned int      primIdx,
1272                                                                    unsigned int      sbtGASIndex,
1273                                                                    float             time,
1274                                                                    float3            data[3])
1275 {
1276     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_triangle_vertex_data, "
1277         "(%0, %10, %11, %12);"
1278         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
1279           "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1280         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1281         :);
1282 }
1283
1284
1285 static __forceinline__ __device__ void optixGetTriangleVertexDataFromHandle(OptixTraversableHandle gas,
1286                                                                    unsigned int      primIdx,
1287                                                                    unsigned int      sbtGASIndex,
1288                                                                    float             time,
1289                                                                    float3            data[3])
1290 {
1291     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_triangle_vertex_data_from_handle, "
1292         "(%0, %10, %11, %12);"
1293         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
1294           "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1295         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1296         :);
1297 }
1298
1299 static __forceinline__ __device__ void optixGetTriangleVertexData(float3 data[3])
1300 {
1301     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_triangle_vertex_data_current_hit, "
1302         "();"
1303         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
1304           "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1305         :);
1306 }
1307
1308 static __forceinline__ __device__ void optixHitObjectGetTriangleVertexData(float3 data[3])
1309 {
1310     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_hitobject_get_triangle_vertex_data, "
1311         "();"
1312         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
1313           "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1314         :);
1315 }
1316
1317
1318 static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
1319                                                                    unsigned int      primIdx,
1320                                                                    unsigned int      sbtGASIndex,
1321                                                                    float             time,
1322                                                                    float4            data[2])
1323 {
1324     asm("call (%0, %1, %2, %3, %4, %5, %6, %7), _optix_get_linear_curve_vertex_data, "
1325         "(%8, %9, %10, %11);"
1326         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1327           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
1328         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1329         :);
1330 }
1331
1332 static __forceinline__ __device__ void optixGetLinearCurveVertexDataFromHandle(OptixTraversableHandle
1333 gas,
1334                                                                    unsigned int      primIdx,
1335                                                                    unsigned int      sbtGASIndex,

```

```

1335                                     float          time,
1336                                     float4         data[2])
1337 {
1338     asm("call (%0, %1, %2, %3, %4, %5, %6, %7), _optix_get_linear_curve_vertex_data_from_handle, "
1339         "(%8, %9, %10, %11);"
1340         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1341           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
1342         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1343         :);
1344 }
1345
1346 static __forceinline__ __device__ void optixGetLinearCurveVertexData(float4 data[2])
1347 {
1348     asm("call (%0, %1, %2, %3, %4, %5, %6, %7), _optix_get_linear_curve_vertex_data_current_hit, "
1349         "();"
1350         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1351           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
1352         :);
1353 }
1354
1355 static __forceinline__ __device__ void optixHitObjectGetLinearCurveVertexData(float4 data[2])
1356 {
1357     asm("call (%0, %1, %2, %3, %4, %5, %6, %7), _optix_hitobject_get_linear_curve_vertex_data, "
1358         "();"
1359         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1360           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
1361         :);
1362 }
1363 }
1364
1365 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(OptixTraversableHandle gas,
1366                                     unsigned int    primIdx,
1367                                     unsigned int    sbtGASIndex,
1368                                     float          time,
1369                                     float4         data[3])
1370 {
1371     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),
1372         _optix_get_quadratic_bspline_vertex_data, "
1373         "(%12, %13, %14, %15);"
1374         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1375           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1376           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w)
1377         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1378         :);
1379 }
1380
1381 static __forceinline__ __device__ void
1382 optixGetQuadraticBSplineVertexDataFromHandle(OptixTraversableHandle gas,
1383                                     unsigned int    primIdx,
1384                                     unsigned int    sbtGASIndex,
1385                                     float          time,
1386                                     float4         data[3])
1387 {
1388     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),
1389         _optix_get_quadratic_bspline_vertex_data_from_handle, "
1390         "(%12, %13, %14, %15);"
1391         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1392           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1393           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w)
1394         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1395         :);
1396 }
1397
1398 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(float4 data[3])
1399 {
1400     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),

```

```

_optix_get_quadratic_bspline_vertex_data_current_hit, "
1398     "();";
1399     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1400       "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1401       "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w)
1402     :);
1403 }
1404
1405 static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineVertexData(float4 data[3])
1406 {
1407     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),
_optix_hitobject_get_quadratic_bspline_vertex_data, "
1408         "());";
1409     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1410       "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1411       "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w)
1412     :);
1413 }
1414
1415 static __forceinline__ __device__ void
optixGetQuadraticBSplineRocapsVertexDataFromHandle(OptixTraversableHandle gas,
1416                                                     unsigned int primIdx,
1417                                                     unsigned int
sbtGASIndex,
1418                                                     float time,
1419                                                     float4 data[3])
1420 {
1421     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),
_optix_get_quadratic_bspline_rocaps_vertex_data_from_handle, "
1422         "(%12, %13, %14, %15);";
1423     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1424       "=f"(data[1].y),
1425       "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
1426       "=f"(data[2].w)
1427     : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1428     :);
1429 }
1430
1431 static __forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexData(float4 data[3])
1432 {
1433     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),
_optix_get_quadratic_bspline_rocaps_vertex_data_current_hit, "
1434         "());";
1435     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1436       "=f"(data[1].y),
1437       "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
1438       "=f"(data[2].w)
1439     :);
1440 }
1441
1442 static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineRocapsVertexData(float4 data[3])
1443 {
1444     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),
_optix_hitobject_get_quadratic_bspline_rocaps_vertex_data, "
1445         "());";
1446     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1447       "=f"(data[1].y),
1448       "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
1449       "=f"(data[2].w)
1450     :);
1451 }
1452
1453 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,
1454                                                     unsigned int primIdx,
1455                                                     unsigned int sbtGASIndex,
1456                                                     float time,
1457                                                     float4 data[4])

```

```

1452 {
1453     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1454         "_optix_get_cubic_bspline_vertex_data, "
1455         "(%16, %17, %18, %19);"
1456         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1457           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1458           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
1459           "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
1460         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1461         :);
1462 }
1463
1464 static __forceinline__ __device__ void optixGetCubicBSplineVertexDataFromHandle(OptixTraversableHandle
gas,
1465
1466                                     unsigned int          primIdx,
1467                                     unsigned int          sbtGASIndex,
1468                                     float                time,
1469                                     float4               data[4])
1470 {
1471     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1472         "_optix_get_cubic_bspline_vertex_data_from_handle, "
1473         "(%16, %17, %18, %19);"
1474         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1475           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1476           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
1477           "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
1478         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1479         :);
1480 }
1481
1482 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(float4 data[4])
1483 {
1484     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1485         "_optix_get_cubic_bspline_vertex_data_current_hit, "
1486         "();"
1487         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1488           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1489           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
1490           "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
1491         :);
1492 }
1493
1494 static __forceinline__ __device__ void optixHitObjectGetCubicBSplineVertexData(float4 data[4])
1495 {
1496     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1497         "_optix_hitobject_get_cubic_bspline_vertex_data, "
1498         "();"
1499         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1500           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1501           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
1502           "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
1503         :);
1504 }
1505
1506 static __forceinline__ __device__ void
optixGetCubicBSplineRocapsVertexDataFromHandle(OptixTraversableHandle gas,
1507
1508                                     unsigned int          primIdx,
1509
1510                                     unsigned int          sbtGASIndex,
1511                                     float                time,
1512                                     float4               data[4])
1513 {
1514     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1515         "_optix_get_cubic_bspline_rocaps_vertex_data_from_handle, "
1516         "(%16, %17, %18, %19);"
1517         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1518           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1519           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
1520         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time), "f"(data[0].x), "f"(data[0].y), "f"(data[0].z), "f"(data[0].w), "f"(data[1].x), "f"(data[1].y), "f"(data[1].z), "f"(data[1].w), "f"(data[2].x), "f"(data[2].y), "f"(data[2].z), "f"(data[2].w), "f"(data[3].x), "f"(data[3].y), "f"(data[3].z), "f"(data[3].w)
1521         :);
1522 }

```

```

1516         "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1517         "=f"(data[3].w)
1518         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1519         :);
1520 }
1521 static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexData(float4 data[4])
1522 {
1523     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1524         "_optix_get_cubic_bspline_rocaps_vertex_data_current_hit, "
1525         "());"
1526         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1527           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1528           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1529           "=f"(data[3].w)
1530         :);
1531 }
1532 static __forceinline__ __device__ void optixHitObjectGetCubicBSplineRocapsVertexData(float4 data[4])
1533 {
1534     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1535         "_optix_hitobject_get_cubic_bspline_rocaps_vertex_data, "
1536         "());"
1537         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1538           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1539           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1540           "=f"(data[3].w)
1541         :);
1542 }
1543 static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas,
1544                             unsigned int primIdx,
1545                             unsigned int sbtGASIndex,
1546                             float time,
1547                             float4 data[4])
1548 {
1549     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1550         "_optix_get_catmullrom_vertex_data, "
1551         "(%16, %17, %18, %19);"
1552         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1553           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1554           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1555           "=f"(data[3].w)
1556         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1557         :);
1558 }
1559 static __forceinline__ __device__ void optixGetCatmullRomVertexDataFromHandle(OptixTraversableHandle
1560 gas,
1561                             unsigned int primIdx,
1562                             unsigned int sbtGASIndex,
1563                             float time,
1564                             float4 data[4])
1565 {
1566     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1567         "_optix_get_catmullrom_vertex_data_from_handle, "
1568         "(%16, %17, %18, %19);"
1569         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1570           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1571           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1572           "=f"(data[3].w)
1573         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1574         :);
1575 }
1576 static __forceinline__ __device__ void optixGetCatmullRomVertexData(float4 data[4])
1577 {

```

```

1577     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1578         "_optix_get_catmullrom_vertex_data_current_hit, "
1579         "());"
1580     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1581       "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1582       "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1583       "=f"(data[3].w)
1584     :);
1585 }
1586 static __forceinline__ __device__ void optixHitObjectGetCatmullRomVertexData(float4 data[4])
1587 {
1588     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1589         "_optix_hitobject_get_catmullrom_vertex_data, "
1590         "());"
1591     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1592       "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1593       "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1594       "=f"(data[3].w)
1595     :);
1596 }
1597 static __forceinline__ __device__ void
1598 optixGetCatmullRomRocapsVertexDataFromHandle(OptixTraversableHandle gas,
1599 primIdx,
1600 sbtGASIndex,
1601 data[4])
1602 {
1603     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1604         "_optix_get_catmullrom_rocaps_vertex_data_from_handle, "
1605         "(%16, %17, %18, %19);"
1606     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1607       "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1608       "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1609       "=f"(data[3].w)
1610     : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1611     :);
1612 }
1613 static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexData(float4 data[4])
1614 {
1615     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1616         "_optix_get_catmullrom_rocaps_vertex_data_current_hit, "
1617         "());"
1618     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1619       "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1620       "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1621       "=f"(data[3].w)
1622     :);
1623 }
1624 static __forceinline__ __device__ void optixHitObjectGetCatmullRomRocapsVertexData(float4 data[4])
1625 {
1626     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1627         "_optix_hitobject_get_catmullrom_rocaps_vertex_data, "
1628         "());"
1629     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1630       "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1631       "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1632       "=f"(data[3].w)
1633     :);
1634 }

```

```

1635 static __forceinline__ __device__ void optixGetCubicBezierVertexData(OptixTraversableHandle gas,
1636                                                                    unsigned int      primIdx,
1637                                                                    unsigned int      sbtGASIndex,
1638                                                                    float             time,
1639                                                                    float4            data[4])
1640 {
1641     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1642         "_optix_get_cubic_bezier_vertex_data, "
1643         "(%16, %17, %18, %19);"
1644         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1645           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1646           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1647           "=f"(data[3].w)
1648         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1649         :);
1650 }
1651 static __forceinline__ __device__ void optixGetCubicBezierVertexDataFromHandle(OptixTraversableHandle
gas,
1652                                                                    unsigned int      primIdx,
1653                                                                    unsigned int      sbtGASIndex,
1654                                                                    float             time,
1655                                                                    float4            data[4])
1656 {
1657     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1658         "_optix_get_cubic_bezier_vertex_data_from_handle, "
1659         "(%16, %17, %18, %19);"
1660         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1661           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1662           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1663           "=f"(data[3].w)
1664         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1665         :);
1666 }
1667 static __forceinline__ __device__ void optixGetCubicBezierVertexData(float4 data[4])
1668 {
1669     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1670         "_optix_get_cubic_bezier_vertex_data_current_hit, "
1671         "();"
1672         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1673           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1674           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1675           "=f"(data[3].w)
1676         :);
1677 }
1678 static __forceinline__ __device__ void optixHitObjectGetCubicBezierVertexData(float4 data[4])
1679 {
1680     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1681         "_optix_hitobject_get_cubic_bezier_vertex_data, "
1682         "();"
1683         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1684           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1685           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1686           "=f"(data[3].w)
1687         :);
1688 }
1689 static __forceinline__ __device__ void
optixGetCubicBezierRocapsVertexDataFromHandle(OptixTraversableHandle gas,
1690                                                                    unsigned int
primIdx,
1691                                                                    unsigned int sbtGASIndex,
1692                                                                    float         time,
1693                                                                    float4         data[4])
1694 {

```

```

1695     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1696         "_optix_get_cubic_bezier_rocaps_vertex_data_from_handle, "
1697         "(%16, %17, %18, %19);"
1698         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1699         "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1700         "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1701         "=f"(data[3].w)
1702         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1703         :);
1704 }
1705 static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexData(float4 data[4])
1706 {
1707     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1708         "_optix_get_cubic_bezier_rocaps_vertex_data_current_hit, "
1709         "();"
1710         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1711         "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1712         "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1713         "=f"(data[3].w)
1714         :);
1715 }
1716 static __forceinline__ __device__ void optixHitObjectGetCubicBezierRocapsVertexData(float4 data[4])
1717 {
1718     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11, %12, %13, %14, %15), "
1719         "_optix_hitobject_get_cubic_bezier_rocaps_vertex_data, "
1720         "();"
1721         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1722         "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1723         "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
1724         "=f"(data[3].w)
1725         :);
1726 }
1727 static __forceinline__ __device__ void optixGetRibbonVertexData(OptixTraversableHandle gas,
1728                                                                unsigned int      primIdx,
1729                                                                unsigned int      sbtGASIndex,
1730                                                                float             time,
1731                                                                float4            data[3])
1732 {
1733     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11), _optix_get_ribbon_vertex_data, "
1734         "(%12, %13, %14, %15);"
1735         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1736         "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
1737         "=f"(data[2].w)
1738         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1739         :);
1740 }
1741 static __forceinline__ __device__ void optixGetRibbonVertexDataFromHandle(OptixTraversableHandle gas,
1742                                                                unsigned int      primIdx,
1743                                                                unsigned int      sbtGASIndex,
1744                                                                float             time,
1745                                                                float4            data[3])
1746 {
1747     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),
1748         _optix_get_ribbon_vertex_data_from_handle, "
1749         "(%12, %13, %14, %15);"
1750         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1751         "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
1752         "=f"(data[2].w)
1753         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1754         :);
1755 }

```

```

1754
1755 static __forceinline__ __device__ void optixGetRibbonVertexData(float4 data[3])
1756 {
1757     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),
1758         _optix_get_ribbon_vertex_data_current_hit, "
1759         "();");
1760     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1761     "=f"(data[1].y),
1762     "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
1763     "=f"(data[2].w)
1764     :);
1765 }
1766
1767 static __forceinline__ __device__ void optixHitObjectGetRibbonVertexData(float4 data[3])
1768 {
1769     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10, %11),
1770         _optix_hitobject_get_ribbon_vertex_data, "
1771         "();");
1772     : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1773     "=f"(data[1].y),
1774     "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
1775     "=f"(data[2].w)
1776     :);
1777 }
1778
1779 static __forceinline__ __device__ float3 optixGetRibbonNormal(OptixTraversableHandle gas,
1780                                                             unsigned int      primIdx,
1781                                                             unsigned int      sbtGASIndex,
1782                                                             float              time,
1783                                                             float2             ribbonParameters)
1784 {
1785     float3 normal;
1786     asm("call (%0, %1, %2), _optix_get_ribbon_normal, "
1787         "(%3, %4, %5, %6, %7, %8);");
1788     : "=f"(normal.x), "=f"(normal.y), "=f"(normal.z)
1789     : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time),
1790       "f"(ribbonParameters.x), "f"(ribbonParameters.y)
1791     :);
1792     return normal;
1793 }
1794
1795 static __forceinline__ __device__ float3 optixGetRibbonNormalFromHandle(OptixTraversableHandle gas,
1796                                                             unsigned int      primIdx,
1797                                                             unsigned int      sbtGASIndex,
1798                                                             float              time,
1799                                                             float2             ribbonParameters)
1800 {
1801     float3 normal;
1802     asm("call (%0, %1, %2), _optix_get_ribbon_normal_from_handle, "
1803         "(%3, %4, %5, %6, %7, %8);");
1804     : "=f"(normal.x), "=f"(normal.y), "=f"(normal.z)
1805     : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time),
1806       "f"(ribbonParameters.x), "f"(ribbonParameters.y)
1807     :);
1808     return normal;
1809 }
1810
1811 static __forceinline__ __device__ float3 optixGetRibbonNormal(float2 ribbonParameters)
1812 {
1813     float3 normal;
1814     asm("call (%0, %1, %2), _optix_get_ribbon_normal_current_hit, "
1815         "(%3, %4);");
1816     : "=f"(normal.x), "=f"(normal.y), "=f"(normal.z)
1817     : "f"(ribbonParameters.x), "f"(ribbonParameters.y)
1818     :);
1819     return normal;
1820 }

```

```

1815
1816 static __forceinline__ __device__ float3 optixHitObjectGetRibbonNormal(float2 ribbonParameters)
1817 {
1818     float3 normal;
1819     asm("call (%0, %1, %2), _optix_hitobject_get_ribbon_normal, "
1820         "(%3, %4);"
1821         : "=f"(normal.x), "=f"(normal.y), "=f"(normal.z)
1822         : "f"(ribbonParameters.x), "f"(ribbonParameters.y)
1823         :);
1824     return normal;
1825 }
1826
1827 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas,
1828                                                         unsigned int      primIdx,
1829                                                         unsigned int      sbtGASIndex,
1830                                                         float             time,
1831                                                         float4            data[1])
1832 {
1833     asm("call (%0, %1, %2, %3), "
1834         "_optix_get_sphere_data, "
1835         "(%4, %5, %6, %7);"
1836         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
1837         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1838         :);
1839 }
1840
1841 static __forceinline__ __device__ void optixGetSphereDataFromHandle(OptixTraversableHandle gas,
1842                                                         unsigned int      primIdx,
1843                                                         unsigned int      sbtGASIndex,
1844                                                         float             time,
1845                                                         float4            data[1])
1846 {
1847     asm("call (%0, %1, %2, %3), "
1848         "_optix_get_sphere_data_from_handle, "
1849         "(%4, %5, %6, %7);"
1850         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
1851         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1852         :);
1853 }
1854
1855 static __forceinline__ __device__ void optixGetSphereData(float4 data[1])
1856 {
1857     asm("call (%0, %1, %2, %3), "
1858         "_optix_get_sphere_data_current_hit, "
1859         "();"
1860         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
1861         :);
1862 }
1863
1864 static __forceinline__ __device__ void optixHitObjectGetSphereData(float4 data[1])
1865 {
1866     asm("call (%0, %1, %2, %3), "
1867         "_optix_hitobject_get_sphere_data, "
1868         "();"
1869         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
1870         :);
1871 }
1872
1873 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle()
1874 {
1875     unsigned long long handle;
1876     asm("call (%0), _optix_get_gas_traversable_handle, ();" : "=l"(handle) :);
1877     return (OptixTraversableHandle)handle;
1878 }
1879
1880 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle handle)
1881 {

```

```

1882     float f0;
1883     asm("call (%0), _optix_get_gas_motion_time_begin, (%1);" : "=f"(f0) : "l"(handle) :);
1884     return f0;
1885 }
1886
1887 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle handle)
1888 {
1889     float f0;
1890     asm("call (%0), _optix_get_gas_motion_time_end, (%1);" : "=f"(f0) : "l"(handle) :);
1891     return f0;
1892 }
1893
1894 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle handle)
1895 {
1896     unsigned int u0;
1897     asm("call (%0), _optix_get_gas_motion_step_count, (%1);" : "=r"(u0) : "l"(handle) :);
1898     return u0;
1899 }
1900
1901 template<typename HitState>
1902 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(const HitState& hs, float
m[12])
1903 {
1904     if(hs.getTransformListSize() == 0)
1905     {
1906         m[0] = 1.0f;
1907         m[1] = 0.0f;
1908         m[2] = 0.0f;
1909         m[3] = 0.0f;
1910         m[4] = 0.0f;
1911         m[5] = 1.0f;
1912         m[6] = 0.0f;
1913         m[7] = 0.0f;
1914         m[8] = 0.0f;
1915         m[9] = 0.0f;
1916         m[10] = 1.0f;
1917         m[11] = 0.0f;
1918         return;
1919     }
1920
1921     float4 m0, m1, m2;
1922     optix_impl::optixGetWorldToObjectTransformMatrix(hs, m0, m1, m2);
1923     m[0] = m0.x;
1924     m[1] = m0.y;
1925     m[2] = m0.z;
1926     m[3] = m0.w;
1927     m[4] = m1.x;
1928     m[5] = m1.y;
1929     m[6] = m1.z;
1930     m[7] = m1.w;
1931     m[8] = m2.x;
1932     m[9] = m2.y;
1933     m[10] = m2.z;
1934     m[11] = m2.w;
1935 }
1936
1937 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12])
1938 {
1939     optixGetWorldToObjectTransformMatrix(OptixIncomingHitObject{}, m);
1940 }
1941
1942 static __forceinline__ __device__ void optixHitObjectGetWorldToObjectTransformMatrix(float m[12])
1943 {
1944     optixGetWorldToObjectTransformMatrix(OptixOutgoingHitObject{}, m);
1945 }
1946
1947 template<typename HitState>

```

```

1948 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(const HitState& hs, float
m[12])
1949 {
1950     if(hs.getTransformListSize() == 0)
1951     {
1952         m[0] = 1.0f;
1953         m[1] = 0.0f;
1954         m[2] = 0.0f;
1955         m[3] = 0.0f;
1956         m[4] = 0.0f;
1957         m[5] = 1.0f;
1958         m[6] = 0.0f;
1959         m[7] = 0.0f;
1960         m[8] = 0.0f;
1961         m[9] = 0.0f;
1962         m[10] = 1.0f;
1963         m[11] = 0.0f;
1964         return;
1965     }
1966
1967     float4 m0, m1, m2;
1968     optix_impl::optixGetObjectToWorldTransformMatrix(hs, m0, m1, m2);
1969     m[0] = m0.x;
1970     m[1] = m0.y;
1971     m[2] = m0.z;
1972     m[3] = m0.w;
1973     m[4] = m1.x;
1974     m[5] = m1.y;
1975     m[6] = m1.z;
1976     m[7] = m1.w;
1977     m[8] = m2.x;
1978     m[9] = m2.y;
1979     m[10] = m2.z;
1980     m[11] = m2.w;
1981 }
1982
1983 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12])
1984 {
1985     optixGetObjectToWorldTransformMatrix(OptixIncomingHitObject{}, m);
1986 }
1987
1988 static __forceinline__ __device__ void optixHitObjectGetObjectToWorldTransformMatrix(float m[12])
1989 {
1990     optixGetObjectToWorldTransformMatrix(OptixOutgoingHitObject{}, m);
1991 }
1992
1993 template<typename HitState>
1994 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(const HitState& hs,
float3 point)
1995 {
1996     if(hs.getTransformListSize() == 0)
1997         return point;
1998
1999     float4 m0, m1, m2;
2000     optix_impl::optixGetWorldToObjectTransformMatrix(hs, m0, m1, m2);
2001     return optix_impl::optixTransformPoint(m0, m1, m2, point);
2002 }
2003
2004 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point)
2005 {
2006     return optixTransformPointFromWorldToObjectSpace(OptixIncomingHitObject{}, point);
2007 }
2008
2009 static __forceinline__ __device__ float3 optixHitObjectTransformPointFromWorldToObjectSpace(float3
point)
2010 {
2011     return optixTransformPointFromWorldToObjectSpace(OptixOutgoingHitObject{}, point);

```

```

2012 }
2013
2014 template<typename HitState>
2015 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(const HitState& hs,
float3 vec)
2016 {
2017     if(hs.getTransformListSize() == 0)
2018         return vec;
2019
2020     float4 m0, m1, m2;
2021     optix_impl::optixGetWorldToObjectTransformMatrix(hs, m0, m1, m2);
2022     return optix_impl::optixTransformVector(m0, m1, m2, vec);
2023 }
2024
2025 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec)
2026 {
2027     return optixTransformVectorFromWorldToObjectSpace(OptixIncomingHitObject{}, vec);
2028 }
2029
2030 static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromWorldToObjectSpace(float3 vec)
2031 {
2032     return optixTransformVectorFromWorldToObjectSpace(OptixOutgoingHitObject{}, vec);
2033 }
2034
2035 template<typename HitState>
2036 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(const HitState& hs,
float3 normal)
2037 {
2038     if(hs.getTransformListSize() == 0)
2039         return normal;
2040
2041     float4 m0, m1, m2;
2042     optix_impl::optixGetObjectToWorldTransformMatrix(hs, m0, m1, m2); // inverse of
optixGetWorldToObjectTransformMatrix()
2043     return optix_impl::optixTransformNormal(m0, m1, m2, normal);
2044 }
2045
2046 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal)
2047 {
2048     return optixTransformNormalFromWorldToObjectSpace(OptixIncomingHitObject{}, normal);
2049 }
2050
2051 static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromWorldToObjectSpace(float3
normal)
2052 {
2053     return optixTransformNormalFromWorldToObjectSpace(OptixOutgoingHitObject{}, normal);
2054 }
2055
2056 template<typename HitState>
2057 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(const HitState& hs,
float3 point)
2058 {
2059     if(hs.getTransformListSize() == 0)
2060         return point;
2061
2062     float4 m0, m1, m2;
2063     optix_impl::optixGetObjectToWorldTransformMatrix(hs, m0, m1, m2);
2064     return optix_impl::optixTransformPoint(m0, m1, m2, point);
2065 }
2066
2067 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point)
2068 {
2069     return optixTransformPointFromObjectToWorldSpace(OptixIncomingHitObject{}, point);
2070 }
2071
2072 static __forceinline__ __device__ float3 optixHitObjectTransformPointFromObjectToWorldSpace(float3
point)

```

```

2073 {
2074     return optixTransformPointFromObjectToWorldSpace(OptixOutgoingHitObject{}, point);
2075 }
2076
2077 template<typename HitState>
2078 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(const HitState& hs,
float3 vec)
2079 {
2080     if(hs.getTransformListSize() == 0)
2081         return vec;
2082
2083     float4 m0, m1, m2;
2084     optix_impl::optixGetObjectToWorldTransformMatrix(hs, m0, m1, m2);
2085     return optix_impl::optixTransformVector(m0, m1, m2, vec);
2086 }
2087
2088 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec)
2089 {
2090     return optixTransformVectorFromObjectToWorldSpace(OptixIncomingHitObject{}, vec);
2091 }
2092
2093 static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromObjectToWorldSpace(float3 vec)
2094 {
2095     return optixTransformVectorFromObjectToWorldSpace(OptixOutgoingHitObject{}, vec);
2096 }
2097
2098 template<typename HitState>
2099 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(const HitState& hs,
float3 normal)
2100 {
2101     if(hs.getTransformListSize() == 0)
2102         return normal;
2103
2104     float4 m0, m1, m2;
2105     optix_impl::optixGetWorldToObjectTransformMatrix(hs, m0, m1, m2); // inverse of
optixGetObjectToWorldTransformMatrix()
2106     return optix_impl::optixTransformNormal(m0, m1, m2, normal);
2107 }
2108
2109 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal)
2110 {
2111     return optixTransformNormalFromObjectToWorldSpace(OptixIncomingHitObject{}, normal);
2112 }
2113
2114 static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromObjectToWorldSpace(float3
normal)
2115 {
2116     return optixTransformNormalFromObjectToWorldSpace(OptixOutgoingHitObject{}, normal);
2117 }
2118
2119 static __forceinline__ __device__ unsigned int optixGetTransformListSize()
2120 {
2121     unsigned int u0;
2122     asm("call (%0), _optix_get_transform_list_size, ();" : "=r"(u0) :);
2123     return u0;
2124 }
2125
2126 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int index)
2127 {
2128     unsigned long long u0;
2129     asm("call (%0), _optix_get_transform_list_handle, (%1);" : "=l"(u0) : "r"(index) :);
2130     return u0;
2131 }
2132
2133 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle)
2134 {

```

```

2135     int i0;
2136     asm("call (%0), _optix_get_transform_type_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
2137     return (OptixTransformType)i0;
2138 }
2139
2140 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle)
2141 {
2142     unsigned long long ptr;
2143     asm("call (%0), _optix_get_static_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
2144     return (const OptixStaticTransform*)ptr;
2145 }
2146
2147 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle)
2148 {
2149     unsigned long long ptr;
2150     asm("call (%0), _optix_get_srt_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
2151     return (const OptixSRTMotionTransform*)ptr;
2152 }
2153
2154 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle)
2155 {
2156     unsigned long long ptr;
2157     asm("call (%0), _optix_get_matrix_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
2158     return (const OptixMatrixMotionTransform*)ptr;
2159 }
2160
2161 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle)
2162 {
2163     int i0;
2164     asm("call (%0), _optix_get_instance_id_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
2165     return i0;
2166 }
2167
2168 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle(OptixTraversableHandle handle)
2169 {
2170     unsigned long long i0;
2171     asm("call (%0), _optix_get_instance_child_from_handle, (%1);" : "=l"(i0) : "l"(handle) :);
2172     return (OptixTraversableHandle)i0;
2173 }
2174
2175 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle)
2176 {
2177     unsigned long long ptr;
2178     asm("call (%0), _optix_get_instance_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
2179     return (const float4*)ptr;
2180 }
2181
2182 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle)
2183 {
2184     unsigned long long ptr;
2185     asm("call (%0), _optix_get_instance_inverse_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle)
:);
2186     return (const float4*)ptr;
2187 }
2188
2189 static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle(OptixTraversableHandle
handle)
2190 {
2191     unsigned long long ptr;
2192     asm("call (%0), _optix_get_gas_ptr_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);

```

```

2193     return (CUDevicePtr)ptr;
2194 }
2195 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind)
2196 {
2197     int ret;
2198     asm volatile(
2199         "call (%0), _optix_report_intersection_0"
2200         ", (%1, %2);"
2201         : "=r"(ret)
2202         : "f"(hitT), "r"(hitKind)
2203         :);
2204     return ret;
2205 }
2206
2207 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
2208 unsigned int a0)
2209 {
2210     int ret;
2211     asm volatile(
2212         "call (%0), _optix_report_intersection_1"
2213         ", (%1, %2, %3);"
2214         : "=r"(ret)
2215         : "f"(hitT), "r"(hitKind), "r"(a0)
2216         :);
2217     return ret;
2218 }
2219
2219 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
2220 unsigned int a0, unsigned int a1)
2221 {
2222     int ret;
2223     asm volatile(
2224         "call (%0), _optix_report_intersection_2"
2225         ", (%1, %2, %3, %4);"
2226         : "=r"(ret)
2227         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1)
2228         :);
2229     return ret;
2230 }
2231
2231 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
2232 unsigned int a0, unsigned int a1, unsigned int a2)
2233 {
2234     int ret;
2235     asm volatile(
2236         "call (%0), _optix_report_intersection_3"
2237         ", (%1, %2, %3, %4, %5);"
2238         : "=r"(ret)
2239         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2)
2240         :);
2241     return ret;
2242 }
2243
2243 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
2244 unsigned int hitKind,
2245 unsigned int a0,
2246 unsigned int a1,
2247 unsigned int a2,
2248 unsigned int a3)
2249 {
2250     int ret;
2251     asm volatile(
2252         "call (%0), _optix_report_intersection_4"
2253         ", (%1, %2, %3, %4, %5, %6);"
2254         : "=r"(ret)
2255         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3)
2256         :);

```

```

2257     return ret;
2258 }
2259
2260 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
2261                                                                unsigned int hitKind,
2262                                                                unsigned int a0,
2263                                                                unsigned int a1,
2264                                                                unsigned int a2,
2265                                                                unsigned int a3,
2266                                                                unsigned int a4)
2267 {
2268     int ret;
2269     asm volatile(
2270         "call (%0), _optix_report_intersection_5"
2271         ", (%1, %2, %3, %4, %5, %6, %7);"
2272         : "=r"(ret)
2273         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4)
2274         :);
2275     return ret;
2276 }
2277
2278 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
2279                                                                unsigned int hitKind,
2280                                                                unsigned int a0,
2281                                                                unsigned int a1,
2282                                                                unsigned int a2,
2283                                                                unsigned int a3,
2284                                                                unsigned int a4,
2285                                                                unsigned int a5)
2286 {
2287     int ret;
2288     asm volatile(
2289         "call (%0), _optix_report_intersection_6"
2290         ", (%1, %2, %3, %4, %5, %6, %7, %8);"
2291         : "=r"(ret)
2292         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5)
2293         :);
2294     return ret;
2295 }
2296
2297 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
2298                                                                unsigned int hitKind,
2299                                                                unsigned int a0,
2300                                                                unsigned int a1,
2301                                                                unsigned int a2,
2302                                                                unsigned int a3,
2303                                                                unsigned int a4,
2304                                                                unsigned int a5,
2305                                                                unsigned int a6)
2306 {
2307     int ret;
2308     asm volatile(
2309         "call (%0), _optix_report_intersection_7"
2310         ", (%1, %2, %3, %4, %5, %6, %7, %8, %9);"
2311         : "=r"(ret)
2312         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6)
2313         :);
2314     return ret;
2315 }
2316
2317 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
2318                                                                unsigned int hitKind,
2319                                                                unsigned int a0,
2320                                                                unsigned int a1,
2321                                                                unsigned int a2,
2322                                                                unsigned int a3,
2323                                                                unsigned int a4,

```

```

2324                                     unsigned int a5,
2325                                     unsigned int a6,
2326                                     unsigned int a7)
2327 {
2328     int ret;
2329     asm volatile(
2330         "call (%0), _optix_report_intersection_8"
2331         ", (%1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
2332         : "=r"(ret)
2333         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6), "r"(a7)
2334         :);
2335     return ret;
2336 }
2337
2338 #define OPTIX_DEFINE_optixGetAttribute_BODY(which)
2339 \
2340 unsigned int ret;
2341 \
2342 asm("call (%0), _optix_get_attribute_" #which ", ();" : "=r"(ret) :);
2343 \
2344     return ret;
2345
2346 static __forceinline__ __device__ unsigned int optixGetAttribute_0()
2347 {
2348     OPTIX_DEFINE_optixGetAttribute_BODY(0);
2349 }
2350
2351 static __forceinline__ __device__ unsigned int optixGetAttribute_1()
2352 {
2353     OPTIX_DEFINE_optixGetAttribute_BODY(1);
2354 }
2355
2356 static __forceinline__ __device__ unsigned int optixGetAttribute_2()
2357 {
2358     OPTIX_DEFINE_optixGetAttribute_BODY(2);
2359 }
2360
2361 static __forceinline__ __device__ unsigned int optixGetAttribute_3()
2362 {
2363     OPTIX_DEFINE_optixGetAttribute_BODY(3);
2364 }
2365
2366 static __forceinline__ __device__ unsigned int optixGetAttribute_4()
2367 {
2368     OPTIX_DEFINE_optixGetAttribute_BODY(4);
2369 }
2370
2371 static __forceinline__ __device__ unsigned int optixGetAttribute_5()
2372 {
2373     OPTIX_DEFINE_optixGetAttribute_BODY(5);
2374 }
2375
2376 static __forceinline__ __device__ unsigned int optixGetAttribute_6()
2377 {
2378     OPTIX_DEFINE_optixGetAttribute_BODY(6);
2379 }
2380
2381 static __forceinline__ __device__ unsigned int optixGetAttribute_7()
2382 {
2383     OPTIX_DEFINE_optixGetAttribute_BODY(7);
2384 }
2385
2386 #undef OPTIX_DEFINE_optixGetAttribute_BODY
2387
2388 static __forceinline__ __device__ void optixTerminateRay()
2389 {
2390     asm volatile("call _optix_terminate_ray, ();");

```

```

2388 }
2389
2390 static __forceinline__ __device__ void optixIgnoreIntersection()
2391 {
2392     asm volatile("call _optix_ignore_intersection, ();");
2393 }
2394
2395 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex()
2396 {
2397     unsigned int u0;
2398     asm("call (%0), _optix_read_primitive_idx, ();" : "=r"(u0) :);
2399     return u0;
2400 }
2401
2402 static __forceinline__ __device__ unsigned int optixGetClusterId()
2403 {
2404     unsigned int u0;
2405     asm("call (%0), _optix_get_cluster_id, ();" : "=r"(u0) :);
2406     return u0;
2407 }
2408
2409 static __forceinline__ __device__ unsigned int optixHitObjectGetClusterId()
2410 {
2411     unsigned int u0;
2412     asm("call (%0), _optix_hitobject_get_cluster_id, ();" : "=r"(u0) :);
2413     return u0;
2414 }
2415
2416 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex()
2417 {
2418     unsigned int u0;
2419     asm("call (%0), _optix_read_sbt_gas_idx, ();" : "=r"(u0) :);
2420     return u0;
2421 }
2422
2423 static __forceinline__ __device__ unsigned int optixGetInstanceId()
2424 {
2425     unsigned int u0;
2426     asm("call (%0), _optix_read_instance_id, ();" : "=r"(u0) :);
2427     return u0;
2428 }
2429
2430 static __forceinline__ __device__ unsigned int optixGetInstanceIndex()
2431 {
2432     unsigned int u0;
2433     asm("call (%0), _optix_read_instance_idx, ();" : "=r"(u0) :);
2434     return u0;
2435 }
2436
2437 static __forceinline__ __device__ unsigned int optixGetHitKind()
2438 {
2439     unsigned int u0;
2440     asm("call (%0), _optix_get_hit_kind, ();" : "=r"(u0) :);
2441     return u0;
2442 }
2443
2444 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind)
2445 {
2446     unsigned int u0;
2447     asm("call (%0), _optix_get_primitive_type_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));
2448     return (OptixPrimitiveType)u0;
2449 }
2450
2451 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind)
2452 {
2453     unsigned int u0;
2454     asm("call (%0), _optix_get_backface_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));

```

```

2455     return (u0 == 0x1);
2456 }
2457
2458 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind)
2459 {
2460     return !optixIsBackFaceHit(hitKind);
2461 }
2462
2463
2464 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType()
2465 {
2466     return optixGetPrimitiveType(optixGetHitKind());
2467 }
2468
2469 static __forceinline__ __device__ bool optixIsBackFaceHit()
2470 {
2471     return optixIsBackFaceHit(optixGetHitKind());
2472 }
2473
2474 static __forceinline__ __device__ bool optixIsFrontFaceHit()
2475 {
2476     return optixIsFrontFaceHit(optixGetHitKind());
2477 }
2478
2479 static __forceinline__ __device__ bool optixIsTriangleHit()
2480 {
2481     return optixIsTriangleFrontFaceHit() || optixIsTriangleBackFaceHit();
2482 }
2483
2484 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit()
2485 {
2486     return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE;
2487 }
2488
2489 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit()
2490 {
2491     return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_BACK_FACE;
2492 }
2493
2494
2495 static __forceinline__ __device__ float optixGetCurveParameter()
2496 {
2497     float f0;
2498     asm("call (%0), _optix_get_curve_parameter, ();" : "=f"(f0) :);
2499     return f0;
2500 }
2501
2502 static __forceinline__ __device__ float optixHitObjectGetCurveParameter()
2503 {
2504     float f0;
2505     asm("call (%0), _optix_hitobject_get_curve_parameter, ();" : "=f"(f0) :);
2506     return f0;
2507 }
2508
2509 static __forceinline__ __device__ float2 optixGetRibbonParameters()
2510 {
2511     float f0, f1;
2512     asm("call (%0, %1), _optix_get_ribbon_parameters, ();" : "=f"(f0), "=f"(f1) :);
2513     return make_float2(f0, f1);
2514 }
2515
2516 static __forceinline__ __device__ float2 optixHitObjectGetRibbonParameters()
2517 {
2518     float f0, f1;
2519     asm("call (%0, %1), _optix_hitobject_get_ribbon_parameters, ();" : "=f"(f0), "=f"(f1) :);
2520     return make_float2(f0, f1);
2521 }

```

```

2522
2523 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics()
2524 {
2525     float f0, f1;
2526     asm("call (%0, %1), _optix_get_triangle_barycentrics, ();" : "=f"(f0), "=f"(f1) :);
2527     return make_float2(f0, f1);
2528 }
2529
2530 static __forceinline__ __device__ float2 optixHitObjectGetTriangleBarycentrics()
2531 {
2532     float f0, f1;
2533     asm("call (%0, %1), _optix_hitobject_get_triangle_barycentrics, ();" : "=f"(f0), "=f"(f1) :);
2534     return make_float2(f0, f1);
2535 }
2536
2537 static __forceinline__ __device__ uint3 optixGetLaunchIndex()
2538 {
2539     unsigned int u0, u1, u2;
2540     asm("call (%0), _optix_get_launch_index_x, ();" : "=r"(u0) :);
2541     asm("call (%0), _optix_get_launch_index_y, ();" : "=r"(u1) :);
2542     asm("call (%0), _optix_get_launch_index_z, ();" : "=r"(u2) :);
2543     return make_uint3(u0, u1, u2);
2544 }
2545
2546 static __forceinline__ __device__ uint3 optixGetLaunchDimensions()
2547 {
2548     unsigned int u0, u1, u2;
2549     asm("call (%0), _optix_get_launch_dimension_x, ();" : "=r"(u0) :);
2550     asm("call (%0), _optix_get_launch_dimension_y, ();" : "=r"(u1) :);
2551     asm("call (%0), _optix_get_launch_dimension_z, ();" : "=r"(u2) :);
2552     return make_uint3(u0, u1, u2);
2553 }
2554
2555 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer()
2556 {
2557     unsigned long long ptr;
2558     asm("call (%0), _optix_get_sbt_data_ptr_64, ();" : "=l"(ptr) :);
2559     return (CUdeviceptr)ptr;
2560 }
2561
2562 static __forceinline__ __device__ void optixThrowException(int exceptionCode)
2563 {
2564     asm volatile(
2565         "call _optix_throw_exception_0, (%0);"
2566         : /* no return value */
2567         : "r"(exceptionCode)
2568         :);
2569 }
2570
2571 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0)
2572 {
2573     asm volatile(
2574         "call _optix_throw_exception_1, (%0, %1);"
2575         : /* no return value */
2576         : "r"(exceptionCode), "r"(exceptionDetail0)
2577         :);
2578 }
2579
2580 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1)
2581 {
2582     asm volatile(
2583         "call _optix_throw_exception_2, (%0, %1, %2);"
2584         : /* no return value */
2585         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1)
2586         :);

```

```

2587 }
2588
2589 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
2590 {
2591     asm volatile(
2592         "call _optix_throw_exception_3, (%0, %1, %2, %3);"
2593         : /* no return value */
2594         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2)
2595         :);
2596 }
2597
2598 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3)
2599 {
2600     asm volatile(
2601         "call _optix_throw_exception_4, (%0, %1, %2, %3, %4);"
2602         : /* no return value */
2603         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3)
2604         :);
2605 }
2606
2607 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3, unsigned int exceptionDetail4)
2608 {
2609     asm volatile(
2610         "call _optix_throw_exception_5, (%0, %1, %2, %3, %4, %5);"
2611         : /* no return value */
2612         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4)
2613         :);
2614 }
2615
2616 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)
2617 {
2618     asm volatile(
2619         "call _optix_throw_exception_6, (%0, %1, %2, %3, %4, %5, %6);"
2620         : /* no return value */
2621         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5)
2622         :);
2623 }
2624
2625 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)
2626 {
2627     asm volatile(
2628         "call _optix_throw_exception_7, (%0, %1, %2, %3, %4, %5, %6, %7);"
2629         : /* no return value */
2630         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6)
2631         :);
2632 }
2633
2634 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)
2635 {
2636     asm volatile(

```

```

2637     "call _optix_throw_exception_8, (%0, %1, %2, %3, %4, %5, %6, %7, %8);"
2638     : /* no return value */
2639     : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6),
"r"(exceptionDetail7)
2640     :);
2641 }
2642
2643 static __forceinline__ __device__ int optixGetExceptionCode()
2644 {
2645     int s0;
2646     asm("call (%0), _optix_get_exception_code, ();" : "=r"(s0) :);
2647     return s0;
2648 }
2649
2650 #define OPTIX_DEFINE_optixGetExceptionDetail_BODY(which)
\
2651 unsigned int ret;
\
2652 asm("call (%0), _optix_get_exception_detail_" #which ", ();" : "=r"(ret) :);
\
2653     return ret;
2654
2655 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0()
2656 {
2657     OPTIX_DEFINE_optixGetExceptionDetail_BODY(0);
2658 }
2659
2660 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1()
2661 {
2662     OPTIX_DEFINE_optixGetExceptionDetail_BODY(1);
2663 }
2664
2665 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2()
2666 {
2667     OPTIX_DEFINE_optixGetExceptionDetail_BODY(2);
2668 }
2669
2670 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3()
2671 {
2672     OPTIX_DEFINE_optixGetExceptionDetail_BODY(3);
2673 }
2674
2675 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4()
2676 {
2677     OPTIX_DEFINE_optixGetExceptionDetail_BODY(4);
2678 }
2679
2680 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5()
2681 {
2682     OPTIX_DEFINE_optixGetExceptionDetail_BODY(5);
2683 }
2684
2685 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6()
2686 {
2687     OPTIX_DEFINE_optixGetExceptionDetail_BODY(6);
2688 }
2689
2690 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7()
2691 {
2692     OPTIX_DEFINE_optixGetExceptionDetail_BODY(7);
2693 }
2694
2695 #undef OPTIX_DEFINE_optixGetExceptionDetail_BODY
2696
2697
2698 static __forceinline__ __device__ char* optixGetExceptionLineInfo()

```

```

2699 {
2700     unsigned long long ptr;
2701     asm("call (%0), _optix_get_exception_line_info, ();" : "=l"(ptr) :);
2702     return (char*)ptr;
2703 }
2704
2705 template <typename ReturnT, typename... ArgTypes>
2706 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args)
2707 {
2708     unsigned long long func;
2709     asm("call (%0), _optix_call_direct_callable, (%1);" : "=l"(func) : "r"(sbtIndex) :);
2710     using funcT = ReturnT (*)(ArgTypes...);
2711     funcT call = (funcT)(func);
2712     return call(args...);
2713 }
2714
2715 template <typename ReturnT, typename... ArgTypes>
2716 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes... args)
2717 {
2718     unsigned long long func;
2719     asm("call (%0), _optix_call_continuation_callable, (%1);" : "=l"(func) : "r"(sbtIndex) :);
2720     using funcT = ReturnT (*)(ArgTypes...);
2721     funcT call = (funcT)(func);
2722     return call(args...);
2723 }
2724
2725 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int
texInfo, float x, float y, unsigned int* singleMipLevel)
2726 {
2727     uint4 result;
2728     unsigned long long resultPtr = reinterpret_cast<unsigned long long>(&result);
2729     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2730     // Cast float args to integers, because the intrinsics take .b32 arguments when compiled to PTX.
2731     asm volatile(
2732         "call _optix_tex_footprint_2d_v2"
2733         ", (%0, %1, %2, %3, %4, %5);"
2734         :
2735         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2736         "l"(singleMipLevelPtr), "l"(resultPtr)
2737         :);
2738     return result;
2739 }
2740
2741 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
2742     unsigned int texInfo,
2743     float x,
2744     float y,
2745     float dPdx_x,
2746     float dPdx_y,
2747     float dPdy_x,
2748     float dPdy_y,
2749     bool coarse,
2750     unsigned int* singleMipLevel)
2751 {
2752     uint4 result;
2753     unsigned long long resultPtr = reinterpret_cast<unsigned long long>(&result);
2754     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2755     // Cast float args to integers, because the intrinsics take .b32 arguments when compiled to PTX.
2756     asm volatile(
2757         "call _optix_tex_footprint_2d_grad_v2"
2758         ", (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
2759         :
2760         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2761         "r"(__float_as_uint(dPdx_x)), "r"(__float_as_uint(dPdx_y)), "r"(__float_as_uint(dPdy_x)),
2762         "r"(__float_as_uint(dPdy_y)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
2763         "l"(resultPtr)
2764         :);

```

```

2764
2765     return result;
2766 }
2767
2768 static __forceinline__ __device__ uint4
2769 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool
coarse, unsigned int* singleMipLevel)
2770 {
2771     uint4          result;
2772     unsigned long long resultPtr          = reinterpret_cast<unsigned long long>(&result);
2773     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2774     // Cast float args to integers, because the intrinsics take .b32 arguments when compiled to PTX.
2775     asm volatile(
2776         "call _optix_tex_footprint_2d_lod_v2"
2777         ", (%0, %1, %2, %3, %4, %5, %6, %7);"
2778         :
2779         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2780         "r"(__float_as_uint(level)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
2781         "l"(resultPtr)
2782         :);
2783     return result;
2784 }
2785 #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)
- template<typename HitState >
static `__forceinline__ __device__ void optix_impl::optixGetWorldToObjectTransformMatrix` (const HitState &hs, float4 &m0, float4 &m1, float4 &m2)
- template<typename HitState >
static `__forceinline__ __device__ void optix_impl::optixGetObjectToWorldTransformMatrix` (const HitState &hs, 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
13 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
14 #error("optix_device_impl_transformations.h is an internal header file and must not be used directly.
15 Please use optix_device.h or optix.h instead.")
16 #endif
17
18 #ifndef OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H
19 #define OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H
20
21 namespace optix_impl {
22
23 static __forceinline__ __device__ float4 optixAddFloat4(const float4& a, const float4& b)
24 {
25     return make_float4(a.x + b.x, a.y + b.y, a.z + b.z, a.w + b.w);
26 }
27
28 static __forceinline__ __device__ float4 optixMulFloat4(const float4& a, float b)
29 {
30     return make_float4(a.x * b, a.y * b, a.z * b, a.w * b);
31 }
32
33 }
34
35 
```

```

39 static __forceinline__ __device__ uint4 optixLdg(unsigned long long addr)
40 {
41     const uint4* ptr;
42     asm volatile("cvta.to.global.u64 %0, %1;" : "=l"(ptr) : "l"(addr));
43     uint4 ret;
44     asm volatile("ld.global.v4.u32 {%0,%1,%2,%3}, [%4];"
45                 : "=r"(ret.x), "=r"(ret.y), "=r"(ret.z), "=r"(ret.w)
46                 : "l"(ptr));
47     return ret;
48 }
49
50 template <class T>
51 static __forceinline__ __device__ T optixLoadReadOnlyAlign16(const T* ptr)
52 {
53     // Debug mode may keep this temporary variable
54     // If T does not enforce 16B alignment, v may not be 16B aligned and storing the loaded data from ptr
55     // fails
56     __align__(16) T v;
57     for(unsigned int ofs = 0; ofs < (unsigned int)sizeof(T); ofs += 16)
58         *(uint4*)((char*)&v + ofs) = optixLdg((unsigned long long)((char*)ptr + ofs));
59     return v;
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 -
126         m1.y * m2.x);
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&
161 m2, const float4* matrix, const float t1)
162 {
163     m0 = optixLoadReadOnlyAlign16(&matrix[0]);
164     m1 = optixLoadReadOnlyAlign16(&matrix[1]);
165     m2 = optixLoadReadOnlyAlign16(&matrix[2]);
166
167     // The conditional prevents concurrent loads leading to spills
168     if(t1 > 0.0f)

```

```

168     {
169         const float t0 = 1.0f - t1;
170         m0 = optixAddFloat4(optixMulFloat4(m0, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[3]),
171         t1));
172         m1 = optixAddFloat4(optixMulFloat4(m1, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[4]),
173         t1));
174         m2 = optixAddFloat4(optixMulFloat4(m2, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[5]),
175         t1));
176     }
177 }
178
179 static __forceinline__ __device__ void optixLoadInterpolatedSrtKey(float4& srt0,
180                             float4& srt1,
181                             float4& srt2,
182                             float4& srt3,
183                             const float4* srt,
184                             const float t1)
185 {
186     srt0 = optixLoadReadOnlyAlign16(&srt[0]);
187     srt1 = optixLoadReadOnlyAlign16(&srt[1]);
188     srt2 = optixLoadReadOnlyAlign16(&srt[2]);
189     srt3 = optixLoadReadOnlyAlign16(&srt[3]);
190
191     // The conditional prevents concurrent loads leading to spills
192     if(t1 > 0.0f)
193     {
194         const float t0 = 1.0f - t1;
195         srt0 = optixAddFloat4(optixMulFloat4(srt0, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[4]),
196         t1));
197         srt1 = optixAddFloat4(optixMulFloat4(srt1, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[5]),
198         t1));
199         srt2 = optixAddFloat4(optixMulFloat4(srt2, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[6]),
200         t1));
201         srt3 = optixAddFloat4(optixMulFloat4(srt3, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[7]),
202         t1));
203
204         float inv_length = 1.f / sqrt(srt2.y * srt2.y + srt2.z * srt2.z + srt2.w * srt2.w + srt3.x *
205         srt3.x);
206         srt2.y *= inv_length;
207         srt2.z *= inv_length;
208         srt2.w *= inv_length;
209         srt3.x *= inv_length;
210     }
211 }
212
213 static __forceinline__ __device__ void optixResolveMotionKey(float& localt, int& key, const
214 OptixMotionOptions& options, const float globalt)
215 {
216     const float timeBegin = options.timeBegin;
217     const float timeEnd = options.timeEnd;
218     const float numIntervals = (float)(options.numKeys - 1);
219
220     // No need to check the motion flags. If data originates from a valid transform list handle, then
221     globalt is in
222     // range, or vanish flags are not set.
223     // should be NaN or in [0,numIntervals]
224     float time = max(0.f, min(numIntervals, numIntervals * __fdividef(globalt - timeBegin, timeEnd -
225     timeBegin)));
226
227     // catch NaN (for example when timeBegin=timeEnd)
228     if(time != time)
229         time = 0.f;
230
231     const float fltKey = fminf(floorf(time), numIntervals - 1);
232
233     localt = time - fltKey;

```

```

224     key    = (int)fltKey;
225 }
226
227 // Returns the interpolated transformation matrix for a particular matrix motion transformation and point
in time.
228 static __forceinline__ __device__ void optixGetInterpolatedTransformation(float4&
trf0,
229                                     float4& trf1,
230                                     float4& trf2,
231                                     const OptixMatrixMotionTransform*
transformData,
232                                     const float time)
233 {
234     // Compute key and intra key time
235     float keyTime;
236     int key;
237     optixResolveMotionKey(keyTime, key, optixLoadReadOnlyAlign16(transformData).motionOptions, time);
238
239     // Get pointer to left key
240     const float4* transform = (const float4*)&transformData->transform[key][0];
241
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) :
optixGetInstanceInverseTransformFromHandle(handle);
305         }
306         else
307         {
308             const OptixStaticTransform* traversable = optixGetStaticTransformFromHandle(handle);
309             transform = (const float4*)((objectToWorld) ? traversable->transform :
traversable->invTransform);
310         }
311
312         trf0 = optixLoadReadOnlyAlign16(&transform[0]);
313         trf1 = optixLoadReadOnlyAlign16(&transform[1]);
314         trf2 = optixLoadReadOnlyAlign16(&transform[2]);
315     }
316     else
317     {
318         trf0 = {1.0f, 0.0f, 0.0f, 0.0f};
319         trf1 = {0.0f, 1.0f, 0.0f, 0.0f};
320         trf2 = {0.0f, 0.0f, 1.0f, 0.0f};
321     }
322 }
323 }
324
325 // Returns the world-to-object transformation matrix resulting from the transform stack and ray time of
the given hit object.
326 template<typename HitState>
327 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(const HitState& hs, float4&
m0, float4& m1, float4& m2)
328 {
329     const unsigned int size = hs.getTransformListSize();
330     const float time = hs.getRayTime();
331
332 #pragma unroll 1
333     for(unsigned int i = 0; i < size; ++i)
334     {
335         OptixTraversableHandle handle = hs.getTransformListHandle(i);
336
337         float4 trf0, trf1, trf2;
338         optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
false);

```

```

339
340     if(i == 0)
341     {
342         m0 = trf0;
343         m1 = trf1;
344         m2 = trf2;
345     }
346     else
347     {
348         // m := trf * m
349         float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
350         m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
351         m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
352         m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
353     }
354 }
355 }
356
357 // Returns the object-to-world transformation matrix resulting from the transform stack and ray time of
358 // the given hit object.
359 template<typename HitState>
360 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(const HitState& hs, float4&
361 m0, float4& m1, float4& m2)
362 {
363     const int size = hs.getTransformListSize();
364     const float time = hs.getRayTime();
365     #pragma unroll 1
366     for(int i = size - 1; i >= 0; --i)
367     {
368         OptixTraversableHandle handle = hs.getTransformListHandle(i);
369
370         float4 trf0, trf1, trf2;
371         optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
372 true);
373
374         if(i == size - 1)
375         {
376             m0 = trf0;
377             m1 = trf1;
378             m2 = trf2;
379         }
380         else
381         {
382             // m := trf * m
383             float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
384             m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
385             m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
386             m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
387         }
388     }
389 }
390
391 // Multiplies the 3x4 matrix with rows m0, m1, m2 with the point p.
392 static __forceinline__ __device__ float3 optixTransformPoint(const float4& m0, const float4& m1, const
393 float4& m2, const float3& p)
394 {
395     float3 result;
396     result.x = m0.x * p.x + m0.y * p.y + m0.z * p.z + m0.w;
397     result.y = m1.x * p.x + m1.y * p.y + m1.z * p.z + m1.w;
398     result.z = m2.x * p.x + m2.y * p.y + m2.z * p.z + m2.w;
399     return result;
400 }
401
402 // Multiplies the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the vector v.
403 static __forceinline__ __device__ float3 optixTransformVector(const float4& m0, const float4& m1, const
404 float4& m2, const float3& v)

```

```

401 {
402     float3 result;
403     result.x = m0.x * v.x + m0.y * v.y + m0.z * v.z;
404     result.y = m1.x * v.x + m1.y * v.y + m1.z * v.z;
405     result.z = m2.x * v.x + m2.y * v.y + m2.z * v.z;
406     return result;
407 }
408
409 // Multiplies the transpose of the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the
normal n.
410 // Note that the given matrix is supposed to be the inverse of the actual transformation matrix.
411 static __forceinline__ __device__ float3 optixTransformNormal(const float4& m0, const float4& m1, const
float4& m2, const float3& n)
412 {
413     float3 result;
414     result.x = m0.x * n.x + m1.x * n.y + m2.x * n.z;
415     result.y = m0.y * n.x + m1.y * n.y + m2.y * n.z;
416     result.z = m0.z * n.x + m1.z * n.y + m2.z * n.z;
417     return result;
418 }
419
420 } // namespace optix_impl
421
422 #endif // 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
32 #ifndef OPTIX_OPTIX_MICROMAP_IMPL_H
33 #define OPTIX_OPTIX_MICROMAP_IMPL_H
34
35 #ifndef OPTIX_MICROMAP_FUNC
36 #ifdef __CUDACC__
37 #define OPTIX_MICROMAP_FUNC __device__
38 #else
39 #define OPTIX_MICROMAP_FUNC
40 #endif
41 #endif
42
43 namespace optix_impl {
44
45 #define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline
46
47 #ifdef __CUDACC__
48 // the device implementation of __uint_as_float is declared in cuda_runtime.h
49 #else
50 // the host implementation of __uint_as_float
51 OPTIX_MICROMAP_INLINE_FUNC float __uint_as_float(unsigned int x)
52 {
53     union { float f; unsigned int i; } var;
54     var.i = x;
55     return var.f;
56 }
57 #endif
58
59 // Extract even bits
60 OPTIX_MICROMAP_INLINE_FUNC unsigned int extractEvenBits(unsigned int x)
61 {
62     x &= 0x55555555;
63     x = (x | (x » 1)) & 0x33333333;
64     x = (x | (x » 2)) & 0x0f0f0f0f;
65 }
66

```

```

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` 90100

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 90100
```

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 */
13
14 #ifndef OPTIX_OPTIX_H
15 #define OPTIX_OPTIX_H
16
17 #define OPTIX_VERSION 90100
18
19 #ifdef __CUDACC__
20 #include "optix_device.h"
21 #else
22 #include "optix_host.h"
23 #endif
24
25 #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
48
49
50
51
52
53
54
55 struct OptixUtilDenoiserImageTile
56 {
57     // input tile image
58     OptixImage2D input;

```

```

59
60     // output tile image
61     OptixImage2D output;
62
63     // overlap offsets, parameters for #optixUtilDenoiserInvoke
64     unsigned int inputOffsetX;
65     unsigned int inputOffsetY;
66 };
67
68 inline OptixResult optixUtilGetPixelStride(const OptixImage2D& image, unsigned int& pixelStrideInBytes)
69 {
70     pixelStrideInBytes = image.pixelStrideInBytes;
71     if(pixelStrideInBytes == 0)
72     {
73         switch(image.format)
74         {
75             case OPTIX_PIXEL_FORMAT_HALF1:
76                 pixelStrideInBytes = 1 * sizeof(short);
77                 break;
78             case OPTIX_PIXEL_FORMAT_HALF2:
79                 pixelStrideInBytes = 2 * sizeof(short);
80                 break;
81             case OPTIX_PIXEL_FORMAT_HALF3:
82                 pixelStrideInBytes = 3 * sizeof(short);
83                 break;
84             case OPTIX_PIXEL_FORMAT_HALF4:
85                 pixelStrideInBytes = 4 * sizeof(short);
86                 break;
87             case OPTIX_PIXEL_FORMAT_FLOAT1:
88                 pixelStrideInBytes = 1 * sizeof(float);
89                 break;
90             case OPTIX_PIXEL_FORMAT_FLOAT2:
91                 pixelStrideInBytes = 2 * sizeof(float);
92                 break;
93             case OPTIX_PIXEL_FORMAT_FLOAT3:
94                 pixelStrideInBytes = 3 * sizeof(float);
95                 break;
96             case OPTIX_PIXEL_FORMAT_FLOAT4:
97                 pixelStrideInBytes = 4 * sizeof(float);
98                 break;
99             case OPTIX_PIXEL_FORMAT_UCHAR3:
100                 pixelStrideInBytes = 3 * sizeof(char);
101                 break;
102             case OPTIX_PIXEL_FORMAT_UCHAR4:
103                 pixelStrideInBytes = 4 * sizeof(char);
104                 break;
105             case OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER:
106                 return OPTIX_ERROR_INVALID_VALUE;
107                 break;
108         }
109     }
110     return OPTIX_SUCCESS;
111 }
112
113 inline OptixResult optixUtilDenoiserSplitImage(
114     const OptixImage2D& input,
115     const OptixImage2D& output,
116     unsigned int overlapWindowSizeInPixels,
117     unsigned int tileWidth,
118     unsigned int tileHeight,
119     std::vector<OptixUtilDenoiserImageTile>& tiles)
120 {
121     if(tileWidth == 0 || tileHeight == 0)
122         return OPTIX_ERROR_INVALID_VALUE;
123
124     unsigned int inPixelStride, outPixelStride;
125     if(const OptixResult res = optixUtilGetPixelStride(input, inPixelStride))

```

```

142         return res;
143     if(const OptixResult res = optixUtilGetPixelStride(output, outPixelStride))
144         return res;
145
146     int inp_w = std::min(tileWidth + 2 * overlapWindowSizeInPixels, input.width);
147     int inp_h = std::min(tileHeight + 2 * overlapWindowSizeInPixels, input.height);
148     int inp_y = 0, copied_y = 0;
149
150     int upscaleX = output.width / input.width;
151     int upscaleY = output.height / input.height;
152
153     do
154     {
155         int inputOffsetY = inp_y == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_h -
156 ((int)input.height - inp_y));
157         int copy_y = inp_y == 0 ? std::min(input.height, tileHeight + overlapWindowSizeInPixels) :
158             std::min(tileHeight, input.height - copied_y);
159
160         int inp_x = 0, copied_x = 0;
161         do
162         {
163             int inputOffsetX = inp_x == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_w -
164 ((int)input.width - inp_x));
165             int copy_x = inp_x == 0 ? std::min(input.width, tileWidth + overlapWindowSizeInPixels) :
166                 std::min(tileWidth, input.width - copied_x);
167
168             OptixUtilDenoiserImageTile tile;
169             tile.input.data = input.data + (size_t)(inp_y - inputOffsetY) *
170 input.rowStrideInBytes
171                 + (size_t)(inp_x - inputOffsetX) * inPixelStride;
172             tile.input.width = inp_w;
173             tile.input.height = inp_h;
174             tile.input.rowStrideInBytes = input.rowStrideInBytes;
175             tile.input.pixelStrideInBytes = input.pixelStrideInBytes;
176             tile.input.format = input.format;
177
178             tile.output.data = output.data + (size_t)(upscaleY * inp_y) *
179 output.rowStrideInBytes
180                 + (size_t)(upscaleX * inp_x) * outPixelStride;
181             tile.output.width = upscaleX * copy_x;
182             tile.output.height = upscaleY * copy_y;
183             tile.output.rowStrideInBytes = output.rowStrideInBytes;
184             tile.output.pixelStrideInBytes = output.pixelStrideInBytes;
185             tile.output.format = output.format;
186
187             tile.inputOffsetX = inputOffsetX;
188             tile.inputOffsetY = inputOffsetY;
189
190             tiles.push_back(tile);
191
192             inp_x += inp_x == 0 ? tileWidth + overlapWindowSizeInPixels : tileWidth;
193             copied_x += copy_x;
194         } while(inp_x < static_cast<int>(input.width));
195
196         inp_y += inp_y == 0 ? tileHeight + overlapWindowSizeInPixels : tileHeight;
197         copied_y += copy_y;
198     } while(inp_y < static_cast<int>(input.height));
199
200     return OPTIX_SUCCESS;
201 }
202
203
204
205 inline OptixResult optixUtilDenoiserInvokeTiled(
206     OptixDenoiser          denoiser,
207     CUstream               stream,
208     const OptixDenoiserParams* params,

```

```

229         CUdeviceptr          denoiserState,
230         size_t                denoiserStateSizeInBytes,
231         const OptixDenoiserGuideLayer* guideLayer,
232         const OptixDenoiserLayer*     layers,
233         unsigned int                numLayers,
234         CUdeviceptr                scratch,
235         size_t                      scratchSizeInBytes,
236         unsigned int                overlapWindowSizeInPixels,
237         unsigned int                tileWidth,
238         unsigned int                tileHeight)
239 {
240     if(!guideLayer || !layers)
241         return OPTIX_ERROR_INVALID_VALUE;
242
243     const unsigned int upscale = numLayers > 0 && layers[0].previousOutput.width == 2 *
layers[0].input.width ? 2 : 1;
244
245     std::vector<std::vector<OptixUtilDenoiserImageTile> tiles(numLayers);
246     std::vector<std::vector<OptixUtilDenoiserImageTile> prevTiles(numLayers);
247     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 = guideLayer->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 = guideLayer->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 = guideLayer->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 = guideLayer->flowTrustworthiness;
298         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->flowTrustworthiness,
299 dummyOutput,
300                                     overlapWindowSizeInPixels,
301                                     tileWidth, tileHeight, flowTrustTiles))
302             return res;
303     }
304     std::vector<OptixUtilDenoiserImageTile> internalGuideLayerTiles;
305     if(guideLayer->previousOutputInternalGuideLayer.data && guideLayer->outputInternalGuideLayer.data)
306     {
307         if(const OptixResult res =
308 optixUtilDenoiserSplitImage(guideLayer->previousOutputInternalGuideLayer,
309                             guideLayer->outputInternalGuideLayer,
310                             upscale * overlapWindowSizeInPixels,
311                             upscale * tileWidth, upscale * tileHeight,
312 internalGuideLayerTiles))
313             return res;
314     }
315     for(size_t t = 0; t < tiles[0].size(); t++)
316     {
317         std::vector<OptixDenoiserLayer> tlayers;
318         for(unsigned int l = 0; l < numLayers; l++)
319         {
320             OptixDenoiserLayer layer = {};
321             layer.input = (tiles[l])[t].input;
322             layer.output = (tiles[l])[t].output;
323             if(layers[l].previousOutput.data)
324                 layer.previousOutput = (prevTiles[l])[t].input;
325             layer.type = layers[l].type;
326             tlayers.push_back(layer);
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

```

```

359 #ifdef __cplusplus
360 }
361 #endif
362
363 #endif // OPTIX_DENOISER_TILING_H

```

8.11 optix_device.h File Reference

Classes

- struct [OptixIncomingHitObject](#)
- struct [OptixOutgoingHitObject](#)
- class [OptixCoopVec](#)< T, N >

Macros

- #define [__OPTIX_INCLUDE_INTERNAL_HEADERS__](#)
- #define [OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET](#)
- #define [OPTIX_INCLUDE_COOPERATIVE_VECTOR](#) 1

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)
- static [__forceinline__ __device__](#) void [optixMakeHitObject](#) ([OptixTraversableHandle](#) handle, float3 rayOrigin, float3 rayDirection, float tmin, float rayTime, unsigned int rayFlags, [OptixTraverseData](#) traverseData, const [OptixTraversableHandle](#) *transforms, unsigned int numTransforms)

- static __forceinline__ __device__ void optixMakeMissHitObject (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int rayFlags)
- static __forceinline__ __device__ void optixMakeNopHitObject ()
- static __forceinline__ __device__ void optixHitObjectGetTraverseData (OptixTraverseData *data)
- 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 optixHitObjectSetSbtRecordIndex (unsigned int sbtRecordIndex)
- static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetGASTraversableHandle ()
- 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__ unsigned int optixGetRemainingTraceDepth ()
- 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 optixHitObjectGetRayFlags ()
- 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 [optixGetTriangleVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])

- static __forceinline__ __device__ void [optixGetTriangleVertexData](#) (float3 data[3])
- static __forceinline__ __device__ void [optixHitObjectGetTriangleVertexData](#) (float3 data[3])
- static __forceinline__ __device__ void [optixGetLinearCurveVertexData](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void [optixGetLinearCurveVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void [optixGetLinearCurveVertexData](#) (float4 data[2])
- static __forceinline__ __device__ void [optixHitObjectGetLinearCurveVertexData](#) (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 [optixGetQuadraticBSplineVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void [optixGetQuadraticBSplineRocapsVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void [optixGetQuadraticBSplineVertexData](#) (float4 data[3])
- static __forceinline__ __device__ void [optixGetQuadraticBSplineRocapsVertexData](#) (float4 data[3])
- static __forceinline__ __device__ void [optixHitObjectGetQuadraticBSplineVertexData](#) (float4 data[3])
- static __forceinline__ __device__ void [optixHitObjectGetQuadraticBSplineRocapsVertexData](#) (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 [optixGetCubicBSplineVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void [optixGetCubicBSplineRocapsVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void [optixGetCubicBSplineVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixGetCubicBSplineRocapsVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixHitObjectGetCubicBSplineVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixHitObjectGetCubicBSplineRocapsVertexData](#) (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 [optixGetCatmullRomVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void [optixGetCatmullRomRocapsVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void [optixGetCatmullRomVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixGetCatmullRomRocapsVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixHitObjectGetCatmullRomVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixHitObjectGetCatmullRomRocapsVertexData](#) (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 [optixGetCubicBezierVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void [optixGetCubicBezierRocapsVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void [optixGetCubicBezierVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixGetCubicBezierRocapsVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixHitObjectGetCubicBezierVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixHitObjectGetCubicBezierRocapsVertexData](#) (float4 data[4])
- static __forceinline__ __device__ void [optixGetRibbonVertexData](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void [optixGetRibbonVertexDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void [optixGetRibbonVertexData](#) (float4 data[3])
- static __forceinline__ __device__ void [optixHitObjectGetRibbonVertexData](#) (float4 data[3])
- static __forceinline__ __device__ float3 [optixGetRibbonNormal](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ float3 [optixGetRibbonNormalFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ float3 [optixGetRibbonNormal](#) (float2 ribbonParameters)
- static __forceinline__ __device__ float3 [optixHitObjectGetRibbonNormal](#) (float2 ribbonParameters)
- static __forceinline__ __device__ void [optixGetSphereData](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ void [optixGetSphereDataFromHandle](#) ([OptixTraversableHandle](#) gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ void [optixGetSphereData](#) (float4 data[1])
- static __forceinline__ __device__ void [optixHitObjectGetSphereData](#) (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__ void optixHitObjectGetWorldToObjectTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixHitObjectGetObjectToWorldTransformMatrix (float m[12])
- static __forceinline__ __device__ float3 optixHitObjectTransformPointFromWorldToObjectSpace (float3 point)
- static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromWorldToObjectSpace (float3 vec)
- static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromWorldToObjectSpace (float3 normal)
- static __forceinline__ __device__ float3 optixHitObjectTransformPointFromObjectToWorldSpace (float3 point)
- static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromObjectToWorldSpace (float3 vec)
- static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromObjectToWorldSpace (float3 normal)
- template<typename HitState >
static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (const HitState &hs, float m[12])
- template<typename HitState >
static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (const HitState &hs, float m[12])
- template<typename HitState >
static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (const HitState &hs, float3 point)
- template<typename HitState >
static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (const HitState &hs, float3 vec)
- template<typename HitState >
static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (const HitState &hs, float3 normal)
- template<typename HitState >
static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (const HitState &hs, float3 point)
- template<typename HitState >
static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (const HitState &hs, float3 vec)
- template<typename HitState >
static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (const HitState &hs, 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 optixGetClusterId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetClusterId ()
- 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__ float2 optixGetTriangleBarycentrics ()
- static __forceinline__ __device__ float2 optixHitObjectGetTriangleBarycentrics ()
- static __forceinline__ __device__ float optixGetCurveParameter ()
- static __forceinline__ __device__ float optixHitObjectGetCurveParameter ()
- static __forceinline__ __device__ float2 optixGetRibbonParameters ()
- static __forceinline__ __device__ float2 optixHitObjectGetRibbonParameters ()
- 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)
- template<typename VecTOut >
static __forceinline__ __device__ VecTOut [optixCoopVecLoad](#) (CUdeviceptr ptr)
- template<typename VecTOut , typename T >
static __forceinline__ __device__ VecTOut [optixCoopVecLoad](#) (T *ptr)
- template<typename VecT >
static __forceinline__ __device__ VecT [optixCoopVecExp2](#) (const VecT &vec)
- template<typename VecT >
static __forceinline__ __device__ VecT [optixCoopVecLog2](#) (const VecT &vec)
- template<typename VecT >
static __forceinline__ __device__ VecT [optixCoopVecTanh](#) (const VecT &vec)
- template<typename VecTOut , typename VecTIn >
static __forceinline__ __device__ VecTOut [optixCoopVecCvt](#) (const VecTIn &vec)
- template<typename VecT >
static __forceinline__ __device__ VecT [optixCoopVecMin](#) (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
static __forceinline__ __device__ VecT [optixCoopVecMin](#) (const VecT &vecA, typename VecT ::value_type B)
- template<typename VecT >
static __forceinline__ __device__ VecT [optixCoopVecMax](#) (const VecT &vecA, const VecT &vecB)

- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecMax (const VecT &vecA, typename VecT ::value_type B)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecMul (const VecT &vecA, const VecT &vecB)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecAdd (const VecT &vecA, const VecT &vecB)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecSub (const VecT &vecA, const VecT &vecB)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecStep (const VecT &vecA, const VecT &vecB)`
- `template<typename VecT >`
`static __forceinline__ __device__ VecT optixCoopVecFFMA (const VecT &vecA, const VecT &vecB, const VecT &vecC)`
- `template<typename VecTOut , typename VecTIn , OptixCoopVecElemType inputInterpretation, OptixCoopVecMatrixLayout matrixLayout, bool transpose, unsigned int N, unsigned int K, OptixCoopVecElemType matrixElementType, OptixCoopVecElemType biasElementType>`
`static __forceinline__ __device__ VecTOut optixCoopVecMatMul (const VecTIn &inputVector, CUdeviceptr matrix, unsigned matrixOffsetInBytes, CUdeviceptr bias, unsigned biasOffsetInBytes, unsigned rowColumnStrideInBytes=0)`
- `template<typename VecTOut , typename VecTIn , OptixCoopVecElemType inputInterpretation, OptixCoopVecMatrixLayout matrixLayout, bool transpose, unsigned int N, unsigned int K, OptixCoopVecElemType matrixElementType>`
`static __forceinline__ __device__ VecTOut optixCoopVecMatMul (const VecTIn &inputVector, CUdeviceptr matrix, unsigned matrixOffsetInBytes, unsigned rowColumnStrideInBytes=0)`
- `template<typename VecTIn >`
`static __forceinline__ __device__ void optixCoopVecReduceSumAccumulate (const VecTIn &inputVector, CUdeviceptr outputVector, unsigned offsetInBytes)`
- `template<typename VecTA , typename VecTB , OptixCoopVecMatrixLayout matrixLayout = OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL>`
`static __forceinline__ __device__ void optixCoopVecOuterProductAccumulate (const VecTA &vecA, const VecTB &vecB, CUdeviceptr outputMatrix, unsigned offsetInBytes, unsigned rowColumnStrideInBytes=0)`
- `template<unsigned int N, unsigned int K, OptixCoopVecElemType elementType, OptixCoopVecMatrixLayout layout = OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCEING_OPTIMAL, unsigned int rowColumnStrideInBytes = 0>`
`static __forceinline__ __device__ unsigned int optixCoopVecGetMatrixSize ()`

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__
```

8.11.2.2 OPTIX_INCLUDE_COOPERATIVE_VECTOR

```
#define OPTIX_INCLUDE_COOPERATIVE_VECTOR 1
```

8.11.2.3 OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET

```
#define OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET
```

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 #ifndef OPTIX_DEVICE_H
15 #define OPTIX_DEVICE_H
16
17 #if defined(__cplusplus) && (__cplusplus < 201103L) && !defined(_WIN32)
18 #error Device code for OptiX requires at least C++11. Consider adding "--std c++11" to the nvcc
19 command-line.
20 #endif
21
22 #include "optix_types.h"
23
24
25 template <typename... Payload>
26 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
27           float3 rayOrigin,
28           float3 rayDirection,
29           float tmin,
30           float tmax,
31           float rayTime,
32           OptixVisibilityMask visibilityMask,
33           unsigned int rayFlags,
34           unsigned int SBTOffset,
35           unsigned int SBTstride,
36           unsigned int missSBTIndex,
37           Payload&... payload);
38
39 template <typename... Payload>
40 static __forceinline__ __device__ void optixTraverse(OptixTraversableHandle handle,
41           float3 rayOrigin,
42           float3 rayDirection,
43           float tmin,
44           float tmax,
45           float rayTime,
46           OptixVisibilityMask visibilityMask,
47           unsigned int rayFlags,
48           unsigned int SBTOffset,
49           unsigned int SBTstride,
50           unsigned int missSBTIndex,
51           Payload&... payload);
52
53 template <typename... Payload>
54 static __forceinline__ __device__ void optixTrace(OptixPayloadTypeID type,
55           OptixTraversableHandle handle,
```

```

118         float3          rayOrigin,
119         float3          rayDirection,
120         float           tmin,
121         float           tmax,
122         float           rayTime,
123         OptixVisibilityMask visibilityMask,
124         unsigned int     rayFlags,
125         unsigned int     SBTOffset,
126         unsigned int     SBTstride,
127         unsigned int     missSBTIndex,
128         Payload&...      payload);
129
130 template <typename... Payload>
131 static __forceinline__ __device__ void optixTraverse(OptixPayloadTypeID type,
132             OptixTraversableHandle handle,
133             float3          rayOrigin,
134             float3          rayDirection,
135             float           tmin,
136             float           tmax,
137             float           rayTime,
138             OptixVisibilityMask visibilityMask,
139             unsigned int     rayFlags,
140             unsigned int     SBTOffset,
141             unsigned int     SBTstride,
142             unsigned int     missSBTIndex,
143             Payload&...      payload);
144
145 static __forceinline__ __device__ void optixReorder(unsigned int coherenceHint, unsigned int
numCoherenceHintBitsFromLSB);
146
147 static __forceinline__ __device__ void optixReorder();
148
149 template <typename... Payload>
150 static __forceinline__ __device__ void optixInvoke(Payload&... payload);
151
152 template <typename... Payload>
153 static __forceinline__ __device__ void optixInvoke(OptixPayloadTypeID type, Payload&... payload);
154
155 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle handle,
156             float3          rayOrigin,
157             float3          rayDirection,
158             float           tmin,
159             float           rayTime,
160             unsigned int     rayFlags,
161             OptixTraverseData traverseData,
162             const OptixTraversableHandle* transforms,
163             unsigned int     numTransforms);
164
165 static __forceinline__ __device__ void optixMakeMissHitObject(unsigned int missSBTIndex,
166             float3          rayOrigin,
167             float3          rayDirection,
168             float           tmin,
169             float           tmax,
170             float           rayTime,
171             unsigned int     rayFlags);
172
173 static __forceinline__ __device__ void optixMakeNopHitObject();
174
175 static __forceinline__ __device__ void optixHitObjectGetTraverseData(OptixTraverseData* data);
176
177 static __forceinline__ __device__ bool optixHitObjectIsHit();
178
179 static __forceinline__ __device__ bool optixHitObjectIsMiss();
180
181 static __forceinline__ __device__ bool optixHitObjectIsNop();
182
183 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex();

```

```
292
296 static __forceinline__ __device__ void optixHitObjectSetSbtRecordIndex(unsigned int sbtRecordIndex);
297
303 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetGASTraversableHandle();
304
310 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p);
311 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p);
312 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p);
313 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p);
314 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p);
315 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p);
316 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p);
317 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p);
318 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p);
319 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p);
320 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p);
321 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p);
322 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p);
323 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p);
324 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p);
325 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p);
326 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p);
327 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p);
328 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p);
329 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p);
330 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p);
331 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p);
332 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p);
333 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p);
334 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p);
335 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p);
336 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p);
337 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p);
338 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p);
339 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p);
340 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p);
341 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p);
342
348 static __forceinline__ __device__ unsigned int optixGetPayload_0();
349 static __forceinline__ __device__ unsigned int optixGetPayload_1();
350 static __forceinline__ __device__ unsigned int optixGetPayload_2();
351 static __forceinline__ __device__ unsigned int optixGetPayload_3();
352 static __forceinline__ __device__ unsigned int optixGetPayload_4();
353 static __forceinline__ __device__ unsigned int optixGetPayload_5();
354 static __forceinline__ __device__ unsigned int optixGetPayload_6();
355 static __forceinline__ __device__ unsigned int optixGetPayload_7();
356 static __forceinline__ __device__ unsigned int optixGetPayload_8();
357 static __forceinline__ __device__ unsigned int optixGetPayload_9();
358 static __forceinline__ __device__ unsigned int optixGetPayload_10();
359 static __forceinline__ __device__ unsigned int optixGetPayload_11();
360 static __forceinline__ __device__ unsigned int optixGetPayload_12();
361 static __forceinline__ __device__ unsigned int optixGetPayload_13();
362 static __forceinline__ __device__ unsigned int optixGetPayload_14();
363 static __forceinline__ __device__ unsigned int optixGetPayload_15();
364 static __forceinline__ __device__ unsigned int optixGetPayload_16();
365 static __forceinline__ __device__ unsigned int optixGetPayload_17();
366 static __forceinline__ __device__ unsigned int optixGetPayload_18();
367 static __forceinline__ __device__ unsigned int optixGetPayload_19();
368 static __forceinline__ __device__ unsigned int optixGetPayload_20();
369 static __forceinline__ __device__ unsigned int optixGetPayload_21();
370 static __forceinline__ __device__ unsigned int optixGetPayload_22();
371 static __forceinline__ __device__ unsigned int optixGetPayload_23();
372 static __forceinline__ __device__ unsigned int optixGetPayload_24();
373 static __forceinline__ __device__ unsigned int optixGetPayload_25();
374 static __forceinline__ __device__ unsigned int optixGetPayload_26();
375 static __forceinline__ __device__ unsigned int optixGetPayload_27();
376 static __forceinline__ __device__ unsigned int optixGetPayload_28();
```

```

377 static __forceinline__ __device__ unsigned int optixGetPayload_29();
378 static __forceinline__ __device__ unsigned int optixGetPayload_30();
379 static __forceinline__ __device__ unsigned int optixGetPayload_31();
380
389 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int typeMask);
390
394 static __forceinline__ __device__ unsigned int optixUndefinedValue();
395
404 static __forceinline__ __device__ unsigned int optixGetRemainingTraceDepth();
405
412 static __forceinline__ __device__ float3 optixGetWorldRayOrigin();
413
419 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin();
420
427 static __forceinline__ __device__ float3 optixGetWorldRayDirection();
428
434 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection();
435
439 static __forceinline__ __device__ float3 optixGetObjectRayOrigin();
440
444 static __forceinline__ __device__ float3 optixGetObjectRayDirection();
445
449 static __forceinline__ __device__ float optixGetRayTmin();
450
456 static __forceinline__ __device__ float optixHitObjectGetRayTmin();
457
466 static __forceinline__ __device__ float optixGetRayTmax();
467
476 static __forceinline__ __device__ float optixHitObjectGetRayTmax();
477
483 static __forceinline__ __device__ float optixGetRayTime();
484
490 static __forceinline__ __device__ float optixHitObjectGetRayTime();
491
495 static __forceinline__ __device__ unsigned int optixGetRayFlags();
496
500 static __forceinline__ __device__ unsigned int optixHitObjectGetRayFlags();
501
505 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask();
506
513 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias, unsigned int instIdx);
514
527 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas,
528                                                                    unsigned int      primIdx,
529                                                                    unsigned int      sbtGASIndex,
530                                                                    float            time,
531                                                                    float3           data[3]);
532
546 static __forceinline__ __device__ void optixGetTriangleVertexDataFromHandle(OptixTraversableHandle gas,
547                                                                    unsigned int      primIdx,
548                                                                    unsigned int      sbtGASIndex,
549                                                                    float            time,
550                                                                    float3           data[3]);
551
560 static __forceinline__ __device__ void optixGetTriangleVertexData(float3 data[3]);
561
568 static __forceinline__ __device__ void optixHitObjectGetTriangleVertexData(float3 data[3]);
569
570
586 static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
587                                                                    unsigned int      primIdx,
588                                                                    unsigned int      sbtGASIndex,
589                                                                    float            time,
590                                                                    float4           data[2]);
591
607 static __forceinline__ __device__ void optixGetLinearCurveVertexDataFromHandle(OptixTraversableHandle

```

```

gas,
608                                     unsigned int      primIdx,
609                                     unsigned int      sbtGASIndex,
610                                     float             time,
611                                     float4            data[2]);
612
621 static __forceinline__ __device__ void optixGetLinearCurveVertexData(float4 data[2]);
622
629 static __forceinline__ __device__ void optixHitObjectGetLinearCurveVertexData(float4 data[2]);
630
643 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(OptixTraversableHandle gas,
644                                     unsigned int      primIdx,
645                                     unsigned int      sbtGASIndex,
646                                     float             time,
647                                     float4            data[3]);
648
661 static __forceinline__ __device__ void
optixGetQuadraticBSplineVertexDataFromHandle(OptixTraversableHandle gas,
662                                     unsigned int      primIdx,
663                                     unsigned int      sbtGASIndex,
664                                     float             time,
665                                     float4            data[3]);
666 static __forceinline__ __device__ void
optixGetQuadraticBSplineRocapsVertexDataFromHandle(OptixTraversableHandle gas,
667                                     unsigned int      primIdx,
668                                     unsigned int      sbtGASIndex,
669                                     float             time,
670                                     float4            data[3]);
671
678 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(float4 data[3]);
679 static __forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexData(float4 data[3]);
680
689 static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineVertexData(float4 data[3]);
690 static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineRocapsVertexData(float4 data[3]);
691
707 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,
708                                     unsigned int      primIdx,
709                                     unsigned int      sbtGASIndex,
710                                     float             time,
711                                     float4            data[4]);
712
725 static __forceinline__ __device__ void optixGetCubicBSplineVertexDataFromHandle(OptixTraversableHandle
gas,
726                                     unsigned int      primIdx,
727                                     unsigned int      sbtGASIndex,
728                                     float             time,
729                                     float4            data[4]);
730 static __forceinline__ __device__ void
optixGetCubicBSplineRocapsVertexDataFromHandle(OptixTraversableHandle gas,
731                                     unsigned int      primIdx,
732                                     unsigned int      sbtGASIndex,
733                                     float             time,
734                                     float4            data[4]);
735
742 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(float4 data[4]);
743 static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexData(float4 data[4]);
744
754 static __forceinline__ __device__ void optixHitObjectGetCubicBSplineVertexData(float4 data[4]);
761 static __forceinline__ __device__ void optixHitObjectGetCubicBSplineRocapsVertexData(float4 data[4]);
762
778 static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas,
779                                     unsigned int      primIdx,

```

```

780                                     unsigned int      sbtGASIndex,
781                                     float              time,
782                                     float4             data[4]);
783
796 static __forceinline__ __device__ void optixGetCatmullRomVertexDataFromHandle(OptixTraversableHandle gas,
797                                     unsigned int      primIdx,
798                                     unsigned int      sbtGASIndex,
799                                     float              time,
800                                     float4             data[4]);
801 static __forceinline__ __device__ void
802 optixGetCatmullRomRocapsVertexDataFromHandle(OptixTraversableHandle gas,
803                                     unsigned int      primIdx,
804                                     unsigned int      sbtGASIndex,
805                                     float              time,
806                                     float4             data[4]);
807
813 static __forceinline__ __device__ void optixGetCatmullRomVertexData(float4 data[4]);
814 static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexData(float4 data[4]);
815
825 static __forceinline__ __device__ void optixHitObjectGetCatmullRomVertexData(float4 data[4]);
826 static __forceinline__ __device__ void optixHitObjectGetCatmullRomRocapsVertexData(float4 data[4]);
827
843 static __forceinline__ __device__ void optixGetCubicBezierVertexData(OptixTraversableHandle gas,
844                                     unsigned int      primIdx,
845                                     unsigned int      sbtGASIndex,
846                                     float              time,
847                                     float4             data[4]);
848
861 static __forceinline__ __device__ void optixGetCubicBezierVertexDataFromHandle(OptixTraversableHandle
862 gas,
863                                     unsigned int      primIdx,
864                                     unsigned int      sbtGASIndex,
865                                     float              time,
866                                     float4             data[4]);
867
868 static __forceinline__ __device__ void
869 optixGetCubicBezierRocapsVertexDataFromHandle(OptixTraversableHandle gas,
870                                     unsigned int      primIdx,
871                                     unsigned int      sbtGASIndex,
872                                     float              time,
873                                     float4             data[4]);
874
878 static __forceinline__ __device__ void optixGetCubicBezierVertexData(float4 data[4]);
879 static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexData(float4 data[4]);
880
890 static __forceinline__ __device__ void optixHitObjectGetCubicBezierVertexData(float4 data[4]);
891 static __forceinline__ __device__ void optixHitObjectGetCubicBezierRocapsVertexData(float4 data[4]);
892
908 static __forceinline__ __device__ void optixGetRibbonVertexData(OptixTraversableHandle gas,
909                                     unsigned int      primIdx,
910                                     unsigned int      sbtGASIndex,
911                                     float              time,
912                                     float4             data[3]);
913
926 static __forceinline__ __device__ void optixGetRibbonVertexDataFromHandle(OptixTraversableHandle gas,
927                                     unsigned int      primIdx,
928                                     unsigned int      sbtGASIndex,
929                                     float              time,
930                                     float4             data[3]);
931
938 static __forceinline__ __device__ void optixGetRibbonVertexData(float4 data[3]);
939
949 static __forceinline__ __device__ void optixHitObjectGetRibbonVertexData(float4 data[3]);
950
957 static __forceinline__ __device__ float3 optixGetRibbonNormal(OptixTraversableHandle gas,

```

```

958                                     unsigned int      primIdx,
959                                     unsigned int      sbtGASIndex,
960                                     float              time,
961                                     float2            ribbonParameters);
962
966 static __forceinline__ __device__ float3 optixGetRibbonNormalFromHandle(OptixTraversableHandle gas,
967                                     unsigned int      primIdx,
968                                     unsigned int      sbtGASIndex,
969                                     float              time,
970                                     float2            ribbonParameters);
971
975 static __forceinline__ __device__ float3 optixGetRibbonNormal(float2 ribbonParameters);
976
980 static __forceinline__ __device__ float3 optixHitObjectGetRibbonNormal(float2 ribbonParameters);
981
997 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas,
998                                     unsigned int      primIdx,
999                                     unsigned int      sbtGASIndex,
1000                                    float              time,
1001                                    float4            data[1]);
1002
1018 static __forceinline__ __device__ void optixGetSphereDataFromHandle(OptixTraversableHandle gas,
1019                                     unsigned int      primIdx,
1020                                     unsigned int      sbtGASIndex,
1021                                     float              time,
1022                                     float4            data[1]);
1023
1032 static __forceinline__ __device__ void optixGetSphereData(float4 data[1]);
1033
1040 static __forceinline__ __device__ void optixHitObjectGetSphereData(float4 data[1]);
1041
1046 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle();
1047
1051 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle gas);
1052
1056 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle gas);
1057
1061 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle gas);
1062
1069 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12]);
1070
1077 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12]);
1078
1085 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point);
1086
1093 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec);
1094
1101 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal);
1102
1109 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point);
1110
1117 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec);
1118
1125 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal);
1126
1133 static __forceinline__ __device__ void optixHitObjectGetWorldToObjectTransformMatrix(float m[12]);
1134
1141 static __forceinline__ __device__ void optixHitObjectGetObjectToWorldTransformMatrix(float m[12]);
1142
1149 static __forceinline__ __device__ float3 optixHitObjectTransformPointFromWorldToObjectSpace(float3
point);
1150
1157 static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromWorldToObjectSpace(float3
vec);
1158
1165 static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromWorldToObjectSpace(float3
normal);

```

```

1166
1173 static __forceinline__ __device__ float3 optixHitObjectTransformPointFromObjectToWorldSpace(float3
point);
1174
1181 static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromObjectToWorldSpace(float3
vec);
1182
1189 static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromObjectToWorldSpace(float3
normal);
1190
1205 template <typename HitState>
1206 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(const HitState& hs, float
m[12]);
1207
1214 template <typename HitState>
1215 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(const HitState& hs, float
m[12]);
1216
1223 template <typename HitState>
1224 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(const HitState& hs,
float3 point);
1225
1232 template <typename HitState>
1233 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(const HitState& hs,
float3 vec);
1234
1241 template <typename HitState>
1242 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(const HitState& hs,
float3 normal);
1243
1250 template <typename HitState>
1251 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(const HitState& hs,
float3 point);
1252
1259 template <typename HitState>
1260 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(const HitState& hs,
float3 vec);
1261
1268 template <typename HitState>
1269 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(const HitState& hs,
float3 normal);
1270
1274 static __forceinline__ __device__ unsigned int optixGetTransformListSize();
1275
1284 static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize();
1285
1289 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int
index);
1290
1299 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle(unsigned
int index);
1300
1301 struct OptixIncomingHitObject
1302 {
1303     __forceinline__ __device__ float          getRayTime()const { return optixGetRayTime(); }
1304     __forceinline__ __device__ unsigned int getTransformListSize()const { return
optixGetTransformListSize(); }
1305     __forceinline__ __device__ OptixTraversableHandle getTransformListHandle(unsigned int index)const
1306     {
1307         return optixGetTransformListHandle(index);
1308     }
1309 };
1310
1311 struct OptixOutgoingHitObject
1312 {
1313     __forceinline__ __device__ float          getRayTime()const { return optixHitObjectGetRayTime(); }
1314     __forceinline__ __device__ unsigned int getTransformListSize()const

```

```

1315 {
1316     return optixHitObjectGetTransformListSize();
1317 }
1318 __forceinline__ __device__ OptixTraversableHandle getTransformListHandle(unsigned int index) const
1319 {
1320     return optixHitObjectGetTransformListHandle(index);
1321 }
1322 };
1323
1327 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle);
1328
1334 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle);
1335
1341 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle);
1342
1348 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle);
1349
1355 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle);
1356
1362 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle(OptixTraversableHandle handle);
1363
1369 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle);
1370
1376 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle);
1377
1383 static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle(OptixTraversableHandle
handle);
1407 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind);
1408
1414 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0);
1415
1421 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1);
1422
1428 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1, unsigned int a2);
1429
1435 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
1436 unsigned int hitKind,
1437 unsigned int a0,
1438 unsigned int a1,
1439 unsigned int a2,
1440 unsigned int a3);
1441
1447 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
1448 unsigned int hitKind,
1449 unsigned int a0,
1450 unsigned int a1,
1451 unsigned int a2,
1452 unsigned int a3,
1453 unsigned int a4);
1454
1460 static __forceinline__ __device__ bool optixReportIntersection(float hitT,
1461 unsigned int hitKind,
1462 unsigned int a0,
1463 unsigned int a1,
1464 unsigned int a2,
1465 unsigned int a3,

```

```

1466                                     unsigned int a4,
1467                                     unsigned int a5);
1468
1474 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
1475                                     unsigned int hitKind,
1476                                     unsigned int a0,
1477                                     unsigned int a1,
1478                                     unsigned int a2,
1479                                     unsigned int a3,
1480                                     unsigned int a4,
1481                                     unsigned int a5,
1482                                     unsigned int a6);
1483
1489 static __forceinline__ __device__ bool optixReportIntersection(float      hitT,
1490                                     unsigned int hitKind,
1491                                     unsigned int a0,
1492                                     unsigned int a1,
1493                                     unsigned int a2,
1494                                     unsigned int a3,
1495                                     unsigned int a4,
1496                                     unsigned int a5,
1497                                     unsigned int a6,
1498                                     unsigned int a7);
1499
1504 static __forceinline__ __device__ unsigned int optixGetAttribute_0();
1505 static __forceinline__ __device__ unsigned int optixGetAttribute_1();
1506 static __forceinline__ __device__ unsigned int optixGetAttribute_2();
1507 static __forceinline__ __device__ unsigned int optixGetAttribute_3();
1508 static __forceinline__ __device__ unsigned int optixGetAttribute_4();
1509 static __forceinline__ __device__ unsigned int optixGetAttribute_5();
1510 static __forceinline__ __device__ unsigned int optixGetAttribute_6();
1511 static __forceinline__ __device__ unsigned int optixGetAttribute_7();
1512
1513
1521 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0();
1522 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1();
1523 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2();
1524 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3();
1525 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4();
1526 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5();
1527 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6();
1528 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7();
1529
1533 static __forceinline__ __device__ void optixTerminateRay();
1534
1539 static __forceinline__ __device__ void optixIgnoreIntersection();
1540
1541
1557 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex();
1558
1559
1568 static __forceinline__ __device__ unsigned int optixGetClusterId();
1569
1578 static __forceinline__ __device__ unsigned int optixHitObjectGetClusterId();
1579
1580
1588 static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex();
1589
1597 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex();
1598
1607 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex();
1608
1609
1622 static __forceinline__ __device__ unsigned int optixGetInstanceId();
1623
1632 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId();
1633

```

```

1643 static __forceinline__ __device__ unsigned int optixGetInstanceIndex();
1644
1653 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex();
1654
1662 static __forceinline__ __device__ unsigned int optixGetHitKind();
1663
1671 static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind();
1672
1676 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind);
1677
1681 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind);
1682
1686 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind);
1687
1691 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType();
1692
1696 static __forceinline__ __device__ bool optixIsFrontFaceHit();
1697
1701 static __forceinline__ __device__ bool optixIsBackFaceHit();
1702
1706 static __forceinline__ __device__ bool optixIsTriangleHit();
1707
1711 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit();
1712
1716 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit();
1717
1718
1725 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics();
1726
1734 static __forceinline__ __device__ float2 optixHitObjectGetTriangleBarycentrics();
1735
1740 static __forceinline__ __device__ float optixGetCurveParameter();
1741
1750 static __forceinline__ __device__ float optixHitObjectGetCurveParameter();
1751
1757 static __forceinline__ __device__ float2 optixGetRibbonParameters();
1758
1767 static __forceinline__ __device__ float2 optixHitObjectGetRibbonParameters();
1768
1775 static __forceinline__ __device__ uint3 optixGetLaunchIndex();
1776
1781 static __forceinline__ __device__ uint3 optixGetLaunchDimensions();
1782
1790 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer();
1791
1798 static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer();
1799
1814 static __forceinline__ __device__ void optixThrowException(int exceptionCode);
1815
1821 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0);
1822
1828 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1829                                     unsigned int exceptionDetail0,
1830                                     unsigned int exceptionDetail1);
1831
1837 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1838                                     unsigned int exceptionDetail0,
1839                                     unsigned int exceptionDetail1,
1840                                     unsigned int exceptionDetail2);
1841
1847 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1848                                     unsigned int exceptionDetail0,
1849                                     unsigned int exceptionDetail1,
1850                                     unsigned int exceptionDetail2,
1851                                     unsigned int exceptionDetail3);
1852

```

```

1858 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1859                                                         unsigned int exceptionDetail0,
1860                                                         unsigned int exceptionDetail1,
1861                                                         unsigned int exceptionDetail2,
1862                                                         unsigned int exceptionDetail3,
1863                                                         unsigned int exceptionDetail4);
1864
1870 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1871                                                         unsigned int exceptionDetail0,
1872                                                         unsigned int exceptionDetail1,
1873                                                         unsigned int exceptionDetail2,
1874                                                         unsigned int exceptionDetail3,
1875                                                         unsigned int exceptionDetail4,
1876                                                         unsigned int exceptionDetail5);
1877
1884 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1885                                                         unsigned int exceptionDetail0,
1886                                                         unsigned int exceptionDetail1,
1887                                                         unsigned int exceptionDetail2,
1888                                                         unsigned int exceptionDetail3,
1889                                                         unsigned int exceptionDetail4,
1890                                                         unsigned int exceptionDetail5,
1891                                                         unsigned int exceptionDetail6);
1892
1898 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1899                                                         unsigned int exceptionDetail0,
1900                                                         unsigned int exceptionDetail1,
1901                                                         unsigned int exceptionDetail2,
1902                                                         unsigned int exceptionDetail3,
1903                                                         unsigned int exceptionDetail4,
1904                                                         unsigned int exceptionDetail5,
1905                                                         unsigned int exceptionDetail6,
1906                                                         unsigned int exceptionDetail7);
1907
1911 static __forceinline__ __device__ int optixGetExceptionCode();
1912
1919 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0();
1920
1926 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1();
1927
1933 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2();
1934
1940 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3();
1941
1947 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4();
1948
1954 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5();
1955
1961 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6();
1962
1968 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7();
1969
1970
1981 static __forceinline__ __device__ char* optixGetExceptionLineInfo();
1982
2006 template <typename ReturnT, typename... ArgTypes>
2007 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args);
2008
2009
2032 template <typename ReturnT, typename... ArgTypes>
2033 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes...
args);
2034
2035
2100 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int
texInfo, float x, float y, unsigned int* singleMipLevel);
2101

```

```

2113 static __forceinline__ __device__ uint4
2114 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool
coarse, unsigned int* singleMipLevel);
2115
2130 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
2131                                                                    unsigned int    texInfo,
2132                                                                    float          x,
2133                                                                    float          y,
2134                                                                    float          dPdx_x,
2135                                                                    float          dPdx_y,
2136                                                                    float          dPdy_x,
2137                                                                    float          dPdy_y,
2138                                                                    bool          coarse,
2139                                                                    unsigned int*  singleMipLevel);
2140 // end group optix_device_api
2142
2143 #define __OPTIX_INCLUDE_INTERNAL_HEADERS__
2144
2145 #include "internal/optix_device_impl.h"
2146
2147
2148 // If you manually define OPTIX_INCLUDE_COOPERATIVE_VECTOR to override the default behavior, you must
2149 // set it to 0 or 1 and not simply define it with no value (which will default it have a value of 0).
2150 #ifndef OPTIX_INCLUDE_COOPERATIVE_VECTOR
2151 #   define OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET
2152 #   define OPTIX_INCLUDE_COOPERATIVE_VECTOR 1
2153 #endif
2154
2155 #if OPTIX_INCLUDE_COOPERATIVE_VECTOR
2161
2166 template <typename VecTOut>
2167 static __forceinline__ __device__ VecTOut optixCoopVecLoad(CUdeviceptr ptr);
2172 template <typename VecTOut, typename T>
2173 static __forceinline__ __device__ VecTOut optixCoopVecLoad(T* ptr);
2174
2175
2181 template <typename VecT>
2182 static __forceinline__ __device__ VecT optixCoopVecExp2(const VecT& vec);
2185 template <typename VecT>
2186 static __forceinline__ __device__ VecT optixCoopVecLog2(const VecT& vec);
2189 template <typename VecT>
2190 static __forceinline__ __device__ VecT optixCoopVecTanh(const VecT& vec);
2195 template <typename VecTOut, typename VecTIn>
2196 static __forceinline__ __device__ VecTOut optixCoopVecCvt(const VecTIn& vec);
2199 template <typename VecT>
2200 static __forceinline__ __device__ VecT optixCoopVecMin(const VecT& vecA, const VecT& vecB);
2203 template <typename VecT>
2204 static __forceinline__ __device__ VecT optixCoopVecMin(const VecT& vecA, typename VecT::value_type B);
2207 template <typename VecT>
2208 static __forceinline__ __device__ VecT optixCoopVecMax(const VecT& vecA, const VecT& vecB);
2211 template <typename VecT>
2212 static __forceinline__ __device__ VecT optixCoopVecMax(const VecT& vecA, typename VecT::value_type B);
2215 template <typename VecT>
2216 static __forceinline__ __device__ VecT optixCoopVecMul(const VecT& vecA, const VecT& vecB);
2219 template <typename VecT>
2220 static __forceinline__ __device__ VecT optixCoopVecAdd(const VecT& vecA, const VecT& vecB);
2223 template <typename VecT>
2224 static __forceinline__ __device__ VecT optixCoopVecSub(const VecT& vecA, const VecT& vecB);
2228 template <typename VecT>
2229 static __forceinline__ __device__ VecT optixCoopVecStep(const VecT& vecA, const VecT& vecB);
2232 template <typename VecT>
2233 static __forceinline__ __device__ VecT optixCoopVecFFMA(const VecT& vecA, const VecT& vecB, const VecT&
vecC);
2234
2300 template <
2301     typename VecTOut,
2302     typename VecTIn,

```

```

2303     OptixCoopVecElemType inputInterpretation,
2304     OptixCoopVecMatrixLayout matrixLayout,
2305     bool transpose,
2306     unsigned int N,
2307     unsigned int K,
2308     OptixCoopVecElemType matrixElementType,
2309     OptixCoopVecElemType biasElementType>
2310 static __forceinline__ __device__ VecTOut optixCoopVecMatMul(const VecTIn& inputVector,
2311                                                             CUdeviceptr matrix, // 64 byte aligned,
Array of KxN elements
2312                                                             unsigned matrixOffsetInBytes, // 64 byte
aligned
2313                                                             CUdeviceptr bias, // 16 byte aligned, Array
of N elements
2314                                                             unsigned biasOffsetInBytes, // 16 byte
aligned
2315                                                             unsigned rowColumnStrideInBytes = 0);
2316
2318 template <typename VecTOut, typename VecTIn, OptixCoopVecElemType inputInterpretation,
OptixCoopVecMatrixLayout matrixLayout, bool transpose, unsigned int N, unsigned int K, OptixCoopVecElemType
matrixElementType>
2319 static __forceinline__ __device__ VecTOut optixCoopVecMatMul(const VecTIn& inputVector,
2320                                                             CUdeviceptr matrix, // 64 byte aligned,
Array of KxN elements
2321                                                             unsigned matrixOffsetInBytes, // 64 byte aligned
2322                                                             unsigned rowColumnStrideInBytes = 0);
2323
2340 template <typename VecTIn>
2341 static __forceinline__ __device__ void optixCoopVecReduceSumAccumulate(const VecTIn& inputVector,
2342                                                             CUdeviceptr outputVector,
2343                                                             unsigned offsetInBytes);
2344
2369 template <typename VecTA, typename VecTB, OptixCoopVecMatrixLayout matrixLayout =
OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL>
2370 static __forceinline__ __device__ void optixCoopVecOuterProductAccumulate(const VecTA& vecA,
2371                                                             const VecTB& vecB,
2372                                                             CUdeviceptr outputMatrix,
2373                                                             unsigned offsetInBytes,
2374                                                             unsigned
rowColumnStrideInBytes = 0);
2375
2398 template <unsigned int N, unsigned int K, OptixCoopVecElemType elementType, OptixCoopVecMatrixLayout
layout = OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCE_OPTIMAL, unsigned int rowColumnStrideInBytes = 0>
2399 static __forceinline__ __device__ unsigned int optixCoopVecGetMatrixSize();
2400
2405 template <typename T, unsigned int N>
2406 class OptixCoopVec
2407 {
2408     public:
2409         static const unsigned int size = N;
2410         using value_type = T;
2411
2412         __forceinline__ __device__ OptixCoopVec() {}
2413         __forceinline__ __device__ OptixCoopVec(const value_type& val)
2414         {
2415             for(unsigned int i = 0; i < size; ++i)
2416                 m_data[i] = val;
2417         }
2418         __forceinline__ __device__ const value_type& operator[](unsigned int index) const { return
m_data[index]; }
2419         __forceinline__ __device__ value_type& operator[](unsigned int index) { return m_data[index]; }
2420
2421         __forceinline__ __device__ const value_type* data() const { return m_data; }
2422         __forceinline__ __device__ value_type* data() { return m_data; }
2423
2424     protected:
2425         value_type m_data[size];

```

```

2426 };
2427 // end group optix_device_api
2429
2430 #include "internal/optix_device_impl_coop_vec.h"
2431
2432 #endif // OPTIX_INCLUDE_COOPERATIVE_VECTOR
2433
2434 #ifdef OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET
2435 # undef OPTIX_INCLUDE_COOPERATIVE_VECTOR
2436 # undef OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET
2437 #endif
2438
2439
2440 #endif // OPTIX_OPTIX_DEVICE_H

```

8.13 optix_function_table.h File Reference

Classes

- struct [OptixFunctionTable](#)

Macros

- #define [OPTIX_ABI_VERSION](#) 118
- #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 118
```

The OptiX ABI version.

8.14 optix_function_table.h

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

```

1 /*
2  * SPDX-FileCopyrightText: Copyright (c) 2019 - 2025 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 118
17
18 #ifndef OPTIX_DEFINE_ABI_VERSION_ONLY
19
20 #include "optix_types.h"
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
43     OptixResult (*optixDeviceContextCreate)(CUcontext fromContext, const OptixDeviceContextOptions*
44 options, OptixDeviceContext* context);
45
46     OptixResult (*optixDeviceContextDestroy)(OptixDeviceContext context);
47
48     OptixResult (*optixDeviceContextGetProperty)(OptixDeviceContext context, OptixDeviceProperty
49 property, void* value, size_t sizeInBytes);
50
51     OptixResult (*optixDeviceContextSetLogCallback)(OptixDeviceContext context,
52                                                     OptixLogCallback callbackFunction,
53                                                     void* callbackData,
54                                                     unsigned int callbackLevel);
55
56     OptixResult (*optixDeviceContextSetCacheEnabled)(OptixDeviceContext context, int enabled);
57
58     OptixResult (*optixDeviceContextSetCacheLocation)(OptixDeviceContext context, const char* location);
59
60     OptixResult (*optixDeviceContextSetCacheDatabaseSizes)(OptixDeviceContext context, size_t
61 lowWaterMark, size_t highWaterMark);
62
63     OptixResult (*optixDeviceContextGetCacheEnabled)(OptixDeviceContext context, int* enabled);
64
65     OptixResult (*optixDeviceContextGetCacheLocation)(OptixDeviceContext context, char* location, size_t
66 locationSize);
67
68     OptixResult (*optixDeviceContextGetCacheDatabaseSizes)(OptixDeviceContext context, size_t*
69 lowWaterMark, size_t* highWaterMark);
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

```

```

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
108     OptixResult (*optixModuleCreateWithTasks)(OptixDeviceContext      context,
109                                                const OptixModuleCompileOptions* moduleCompileOptions,
110                                                const OptixPipelineCompileOptions* pipelineCompileOptions,
111                                                const char*                  input,
112                                                size_t                      inputSize,
113                                                char*                      logString,
114                                                size_t*                    logStringSize,
115                                                OptixModule*                module,
116                                                OptixTask*                firstTask);
117
118     OptixResult (*optixModuleGetCompilationState)(OptixModule module, OptixModuleCompileState* state);
119
120     OptixResult (*optixModuleCancelCreation)(OptixModule module, OptixCreationFlags flags);
121
122     OptixResult (*optixStub)(void);
123
124     OptixResult (*optixDeviceContextCancelCreations)(OptixDeviceContext context, OptixCreationFlags
125 flags);
126
127     OptixResult (*optixModuleDestroy)(OptixModule module);
128
129     OptixResult(*optixBuiltinISModuleGet)(OptixDeviceContext      context,
130                                           const OptixModuleCompileOptions* moduleCompileOptions,
131                                           const OptixPipelineCompileOptions* pipelineCompileOptions,
132                                           const OptixBuiltinISOptions*    builtinISOptions,
133                                           OptixModule*                builtinModule);
134
135     //@ }
136     //@ {
137
138     OptixResult (*optixTaskExecute)(OptixTask      task,
139                                     OptixTask*    additionalTasks,
140                                     unsigned int   maxNumAdditionalTasks,
141                                     unsigned int*  numAdditionalTasksCreated);
142
143     OptixResult (*optixTaskGetSerializationKey)(OptixTask task, void* key, size_t* size);
144
145     OptixResult (*optixTaskSerializeOutput)(OptixTask task, void* data, size_t* size);
146
147     OptixResult (*optixTaskDeserializeOutput)(OptixTask      task,
148                                               const void*    data,
149                                               size_t        size,
150                                               OptixTask*    additionalTasks,
151                                               unsigned int   maxNumAdditionalTasks,
152                                               unsigned int*  numAdditionalTasksCreated);
153
154     //@ }
155     //@ {
156
157     OptixResult (*optixProgramGroupCreate)(OptixDeviceContext      context,
158                                             const OptixProgramGroupDesc* programDescriptions,
159                                             unsigned int               numProgramGroups,
160                                             const OptixProgramGroupOptions* options,
161                                             char*                   logString,
162                                             size_t*                logStringSize,
163                                             OptixProgramGroup*    programGroups);
164
165     OptixResult (*optixProgramGroupDestroy)(OptixProgramGroup programGroup);

```

```

179
181     OptixResult (*optixProgramGroupGetStackSize)(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes, OptixPipeline pipeline);
182
183     //@ }
184     //@ {
185
186
187
188     OptixResult (*optixPipelineCreate)(OptixDeviceContext          context,
189                                       const OptixPipelineCompileOptions* pipelineCompileOptions,
190                                       const OptixPipelineLinkOptions*     pipelineLinkOptions,
191                                       const OptixProgramGroup*           programGroups,
192                                       unsigned int                       numProgramGroups,
193                                       char*                               logString,
194                                       size_t*                             logStringSize,
195                                       OptixPipeline*                     pipeline);
196
197
198     OptixResult (*optixPipelineDestroy)(OptixPipeline pipeline);
199
200
201     OptixResult (*optixPipelineSetStackSizeFromCallDepths)(OptixPipeline pipeline,
202                                                            unsigned int maxTraceDepth,
203                                                            unsigned int maxContinuationCallableDepth,
204                                                            unsigned int maxDirectCallableDepthFromState,
205                                                            unsigned int maxDirectCallableDepthFromTraversal,
206                                                            unsigned int maxTraversableGraphDepth);
207
208
209     OptixResult (*optixPipelineSetStackSize)(OptixPipeline pipeline,
210                                              unsigned int directCallableStackSizeFromTraversal,
211                                              unsigned int directCallableStackSizeFromState,
212                                              unsigned int continuationStackSize,
213                                              unsigned int maxTraversableGraphDepth);
214
215
216     OptixResult (*optixPipelineSymbolMemcpyAsync)(OptixPipeline          pipeline,
217                                                    const char*             name,
218                                                    void*                     mem,
219                                                    size_t                   sizeInBytes,
220                                                    size_t                   offsetInBytes,
221                                                    OptixPipelineSymbolMemcpyKind kind,
222                                                    CUstream                  stream);
223
224     //@ }
225     //@ {
226
227
228
229     OptixResult (*optixAccelComputeMemoryUsage)(OptixDeviceContext          context,
230                                                 const OptixAccelBuildOptions* accelOptions,
231                                                 const OptixBuildInput*      buildInputs,
232                                                 unsigned int                numBuildInputs,
233                                                 OptixAccelBufferSizes*    bufferSizes);
234
235
236
237     OptixResult (*optixAccelBuild)(OptixDeviceContext          context,
238                                    CUstream                   stream,
239                                    const OptixAccelBuildOptions* accelOptions,
240                                    const OptixBuildInput*      buildInputs,
241                                    unsigned int                numBuildInputs,
242                                    CUdeviceptr                 tempBuffer,
243                                    size_t                      tempBufferSizeInBytes,
244                                    CUdeviceptr                 outputBuffer,
245                                    size_t                      outputBufferSizeInBytes,
246                                    OptixTraversableHandle*     outputHandle,
247                                    const OptixAccelEmitDesc*    emittedProperties,
248                                    unsigned int                numEmittedProperties);
249
250
251     OptixResult (*optixAccelGetRelocationInfo)(OptixDeviceContext context, OptixTraversableHandle
handle, OptixRelocationInfo* info);
252
253
254     OptixResult (*optixCheckRelocationCompatibility)(OptixDeviceContext          context,
255                                                      const OptixRelocationInfo* info,

```

```

255                                     int*                               compatible);
256
258     OptixResult (*optixAccelRelocate)(OptixDeviceContext      context,
259                                     CUstream                stream,
260                                     const OptixRelocationInfo* info,
261                                     const OptixRelocateInput* relocateInputs,
262                                     size_t                  numRelocateInputs,
263                                     CUdeviceptr             targetAccel,
264                                     size_t                   targetAccelSizeInBytes,
265                                     OptixTraversableHandle* targetHandle);
266
267
269     OptixResult (*optixAccelCompact)(OptixDeviceContext      context,
270                                     CUstream                stream,
271                                     OptixTraversableHandle inputHandle,
272                                     CUdeviceptr             outputBuffer,
273                                     size_t                   outputBufferSizeInBytes,
274                                     OptixTraversableHandle* outputHandle);
275
276     OptixResult (*optixAccelEmitProperty)(OptixDeviceContext context,
277                                     CUstream                stream,
278                                     OptixTraversableHandle handle,
279                                     const OptixAccelEmitDesc* emittedProperty);
280
282     OptixResult (*optixConvertPointerToTraversableHandle)(OptixDeviceContext onDevice,
283                                                         CUdeviceptr pointer,
284                                                         OptixTraversableType traversableType,
285                                                         OptixTraversableHandle* traversableHandle);
286
288     OptixResult (*optixOpacityMicromapArrayComputeMemoryUsage)(OptixDeviceContext
context,
289                                                         const OptixOpacityMicromapArrayBuildInput*
buildInput,
290                                                         OptixMicromapBufferSizes*
bufferSizes);
291
293     OptixResult (*optixOpacityMicromapArrayBuild)(OptixDeviceContext      context,
294                                     CUstream                stream,
295                                     const OptixOpacityMicromapArrayBuildInput* buildInput,
296                                     const OptixMicromapBuffers* buffers);
297
299     OptixResult (*optixOpacityMicromapArrayGetRelocationInfo)(OptixDeviceContext context,
300                                                         CUdeviceptr opacityMicromapArray,
301                                                         OptixRelocationInfo* info);
302
304     OptixResult (*optixOpacityMicromapArrayRelocate)(OptixDeviceContext      context,
305                                     CUstream                stream,
306                                     const OptixRelocationInfo* info,
307                                     CUdeviceptr             targetOpacityMicromapArray,
308                                     size_t                   targetOpacityMicromapArraySizeInBytes);
309
310     OptixResult (*stub1)(void);
311     OptixResult (*stub2)(void);
312
314     OptixResult (*optixClusterAccelComputeMemoryUsage)(OptixDeviceContext      context,
315                                                         OptixClusterAccelBuildMode buildMode,
316                                                         const OptixClusterAccelBuildInput* buildInput,
317                                                         OptixAccelBufferSizes* bufferSizes);
318
320     OptixResult (*optixClusterAccelBuild)(OptixDeviceContext      context,
321                                     CUstream                stream,
322                                     const OptixClusterAccelBuildModeDesc* buildModeDesc,
323                                     const OptixClusterAccelBuildInput* buildInput,
324                                     CUdeviceptr             argsArray,
325                                     CUdeviceptr             argsCount,
326                                     unsigned int             argsStrideInBytes);

```

```

327
328     //@ }
329     //@ {
330
331     OptixResult (*optixSbtRecordPackHeader)(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer);
332
333     OptixResult (*optixLaunch)(OptixPipeline                pipeline,
334                                CUstream                    stream,
335                                CUdeviceptr                  pipelineParams,
336                                size_t                       pipelineParamsSize,
337                                const OptixShaderBindingTable* sbt,
338                                unsigned int                  width,
339                                unsigned int                  height,
340                                unsigned int                  depth);
341
342     //@ }
343     //@ {
344
345     OptixResult (*optixCoopVecMatrixConvert)(OptixDeviceContext context,
346                                              CUstream            stream,
347                                              unsigned int        numNetworks,
348                                              const OptixNetworkDescription* inputNetworkDescription,
349                                              CUdeviceptr          inputNetworks,
350                                              size_t               inputNetworkStrideInBytes,
351                                              const OptixNetworkDescription* outputNetworkDescription,
352                                              CUdeviceptr          outputNetworks,
353                                              size_t               outputNetworkStrideInBytes);
354
355     OptixResult (*optixCoopVecMatrixComputeSize)(OptixDeviceContext context,
356                                                  unsigned int        N,
357                                                  unsigned int        K,
358                                                  OptixCoopVecElemType elementType,
359                                                  OptixCoopVecMatrixLayout layout,
360                                                  size_t               rowColumnStrideInBytes,
361                                                  size_t*              sizeInBytes);
362
363     //@ }
364     //@ {
365
366     OptixResult (*optixDenoiserCreate)(OptixDeviceContext context, OptixDenoiserModelKind modelKind,
367 const OptixDenoiserOptions* options, OptixDenoiser* returnHandle);
368
369     OptixResult (*optixDenoiserDestroy)(OptixDenoiser handle);
370
371     OptixResult (*optixDenoiserComputeMemoryResources)(const OptixDenoiser handle,
372                                                       unsigned int        maximumInputWidth,
373                                                       unsigned int        maximumInputHeight,
374                                                       OptixDenoiserSizes* returnSizes);
375
376     OptixResult (*optixDenoiserSetup)(OptixDenoiser denoiser,
377                                       CUstream        stream,
378                                       unsigned int    inputWidth,
379                                       unsigned int    inputHeight,
380                                       CUdeviceptr     state,
381                                       size_t           stateSizeInBytes,
382                                       CUdeviceptr     scratch,
383                                       size_t           scratchSizeInBytes);
384
385     OptixResult (*optixDenoiserInvoke)(OptixDenoiser denoiser,
386                                       CUstream        stream,
387                                       const OptixDenoiserParams* params,
388                                       CUdeviceptr     denoiserState,
389                                       size_t           denoiserStateSizeInBytes,
390                                       const OptixDenoiserGuideLayer * guideLayer,
391                                       const OptixDenoiserLayer * layers,
392                                       unsigned int     numLayers,
393

```

```

404             unsigned int             inputOffsetX,
405             unsigned int             inputOffsetY,
406             CUdeviceptr              scratch,
407             size_t                   scratchSizeInBytes);
408
410     OptixResult (*optixDenoiserComputeIntensity)(OptixDenoiser handle,
411                                                  CUstream stream,
412                                                  const OptixImage2D* inputImage,
413                                                  CUdeviceptr outputIntensity,
414                                                  CUdeviceptr scratch,
415                                                  size_t scratchSizeInBytes);
416
418     OptixResult (*optixDenoiserComputeAverageColor)(OptixDenoiser handle,
419                                                     CUstream stream,
420                                                     const OptixImage2D* inputImage,
421                                                     CUdeviceptr outputAverageColor,
422                                                     CUdeviceptr scratch,
423                                                     size_t scratchSizeInBytes);
424
426     OptixResult (*optixDenoiserCreateWithUserModel)(OptixDeviceContext context, const void * data, size_t
dataSizeInBytes, OptixDenoiser* returnHandle);
427     //@ }
428
429 } OptixFunctionTable;
430
431 // define global function table variable with ABI specific name.
432 #define OPTIX_CONCATENATE_ABI_VERSION(prefix, macro) OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro)
433 #define OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro) prefix ## _ ## macro
434 #define OPTIX_FUNCTION_TABLE_SYMBOL OPTIX_CONCATENATE_ABI_VERSION(g_optixFunctionTable,
OPTIX_ABI_VERSION)
435 // end group optix_function_table
436
437
438 #ifdef __cplusplus
439 }
440 #endif
441
442 #endif /* OPTIX_DEFINE_ABI_VERSION_ONLY */
443
444 #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 optixPipelineSetStackSizeFromCallDepths` (`OptixPipeline` pipeline, `unsigned int` maxTraceDepth, `unsigned int` maxContinuationCallableDepth, `unsigned int` maxDirectCallableDepthFromState, `unsigned int` maxDirectCallableDepthFromTraversal, `unsigned int` maxTraversableGraphDepth)
- OPTIXAPI `OptixResult optixPipelineSetStackSize` (`OptixPipeline` pipeline, `unsigned int` directCallableStackSizeFromTraversal, `unsigned int` directCallableStackSizeFromState, `unsigned int` continuationStackSize, `unsigned int` maxTraversableGraphDepth)
- OPTIXAPI `OptixResult optixPipelineSymbolMemcpyAsync` (`OptixPipeline` pipeline, `const` `char` *name, `void` *mem, `size_t` sizeInBytes, `size_t` offsetInBytes, `OptixPipelineSymbolMemcpyKind` kind, `CUstream` stream)
- 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 optixModuleCancelCreation` (`OptixModule` module, `OptixCreationFlags` flags)
- OPTIXAPI `OptixResult optixDeviceContextCancelCreations` (`OptixDeviceContext` context, `OptixCreationFlags` flags)
- OPTIXAPI `OptixResult optixModuleDestroy` (`OptixModule` module)
- OPTIXAPI `OptixResult optixBuiltinISModuleGet` (`OptixDeviceContext` context, `const` `OptixModuleCompileOptions` *moduleCompileOptions, `const` `OptixPipelineCompileOptions` *pipelineCompileOptions, `const` `OptixBuiltinISOptions` *builtinISOptions, `OptixModule` *builtinModule)
- OPTIXAPI `OptixResult optixTaskExecute` (`OptixTask` task, `OptixTask` *additionalTasks, `unsigned int` maxNumAdditionalTasks, `unsigned int` *numAdditionalTasksCreated)
- OPTIXAPI `OptixResult optixTaskGetSerializationKey` (`OptixTask` task, `void` *key, `size_t` *size)
- OPTIXAPI `OptixResult optixTaskSerializeOutput` (`OptixTask` task, `void` *data, `size_t` *size)
- OPTIXAPI `OptixResult optixTaskDeserializeOutput` (`OptixTask` task, `const` `void` *data, `size_t` size, `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 optixSbtRecordPackHeader` (`OptixProgramGroup` programGroup, `void *sbtRecordHeaderHostPointer`)
- 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 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 optixClusterAccelComputeMemoryUsage` (`OptixDeviceContext` context, `OptixClusterAccelBuildMode` buildMode, `const OptixClusterAccelBuildInput *buildInput`, `OptixAccelBufferSizes *bufferSizes`)
- OPTIXAPI `OptixResult optixClusterAccelBuild` (`OptixDeviceContext` context, `CUstream` stream, `const OptixClusterAccelBuildModeDesc *buildModeDesc`, `const OptixClusterAccelBuildInput *buildInput`, `CUdeviceptr` argsArray, `CUdeviceptr` argsCount, `unsigned int` argsStrideInBytes)
- OPTIXAPI `OptixResult optixCoopVecMatrixConvert` (`OptixDeviceContext` context, `CUstream` stream, `unsigned int` numNetworks, `const OptixNetworkDescription *inputNetworkDescription`,

- `CUdeviceptr` inputNetworks, `size_t` inputNetworkStrideInBytes, `const OptixNetworkDescription` *outputNetworkDescription, `CUdeviceptr` outputNetworks, `size_t` outputNetworkStrideInBytes)
- **OPTIXAPI** `OptixResult` `optixCoopVecMatrixComputeSize` (`OptixDeviceContext` context, `unsigned int` N, `unsigned int` K, `OptixCoopVecElemType` elementType, `OptixCoopVecMatrixLayout` layout, `size_t` rowColumnStrideInBytes, `size_t` *sizeInBytes)
 - **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`.

8.18 optix_host.h

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

```
1 /*
2  * SPDX-FileCopyrightText: Copyright (c) 2010 - 2025 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
51
55
66 OPTIXAPI const char* optixGetErrorName(OptixResult result);
67
78 OPTIXAPI const char* optixGetErrorString(OptixResult result);
79
84
103 OPTIXAPI OptixResult optixDeviceContextCreate(CUcontext fromContext, const OptixDeviceContextOptions*
options, OptixDeviceContext* context);
104
113 OPTIXAPI OptixResult optixDeviceContextDestroy(OptixDeviceContext context);
114
121 OPTIXAPI OptixResult optixDeviceContextGetProperty(OptixDeviceContext context, OptixDeviceProperty
property, void* value, size_t sizeInBytes);
122
136 OPTIXAPI OptixResult optixDeviceContextSetLogCallback(OptixDeviceContext context,
137                                                         OptixLogCallback callbackFunction,
138                                                         void* callbackData,
139                                                         unsigned int callbackLevel);
140
159 OPTIXAPI OptixResult optixDeviceContextSetCacheEnabled(OptixDeviceContext context, int enabled);
160
181 OPTIXAPI OptixResult optixDeviceContextSetCacheLocation(OptixDeviceContext context, const char*
location);
182
210 OPTIXAPI OptixResult optixDeviceContextSetCacheDatabaseSizes(OptixDeviceContext context, size_t
lowWaterMark, size_t highWaterMark);
211
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
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
287 OPTIXAPI OptixResult optixPipelineSetStackSizeFromCallDepths(OptixPipeline pipeline,
288                                                             unsigned int maxTraceDepth,
289                                                             unsigned int maxContinuationCallableDepth,
290                                                             unsigned int maxDirectCallableDepthFromState,
291                                                             unsigned int maxDirectCallableDepthFromTraversal,
292                                                             unsigned int maxTraversableGraphDepth);
293
316 OPTIXAPI OptixResult optixPipelineSetStackSize(OptixPipeline pipeline,
317                                                 unsigned int directCallableStackSizeFromTraversal,
318                                                 unsigned int directCallableStackSizeFromState,
319                                                 unsigned int continuationStackSize,
320                                                 unsigned int maxTraversableGraphDepth);
321
335 OPTIXAPI OptixResult optixPipelineSymbolMemcpyAsync(OptixPipeline pipeline,
336                                                       const char* name,
337                                                       void* mem,
338                                                       size_t sizeInBytes,
339                                                       size_t offsetInBytes,
340                                                       OptixPipelineSymbolMemcpyKind kind,
341                                                       CUstream stream);
342
347
377 OPTIXAPI OptixResult optixModuleCreate(OptixDeviceContext context,
378                                         const OptixModuleCompileOptions* moduleCompileOptions,
379                                         const OptixPipelineCompileOptions* pipelineCompileOptions,
380                                         const char* input,
381                                         size_t inputSize,
382                                         char* logString,
383                                         size_t* logStringSize,
384                                         OptixModule* module);
385
426 OPTIXAPI 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
442 OPTIXAPI OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState* state);
443
455 OPTIXAPI OptixResult optixModuleCancelCreation(OptixModule module, OptixCreationFlags flags);
456
457
462 OPTIXAPI OptixResult optixDeviceContextCancelCreations(OptixDeviceContext context, OptixCreationFlags
flags);
463
469 OPTIXAPI OptixResult optixModuleDestroy(OptixModule module);

```

```

470
474 OPTIXAPI OptixResult optixBuiltinISModuleGet(OptixDeviceContext      context,
475                                               const OptixModuleCompileOptions* moduleCompileOptions,
476                                               const OptixPipelineCompileOptions* pipelineCompileOptions,
477                                               const OptixBuiltinISOOptions* builtinISOOptions,
478                                               OptixModule* builtinModule);
479
484
502 OPTIXAPI OptixResult optixTaskExecute(OptixTask      task,
503                                       OptixTask*    additionalTasks,
504                                       unsigned int    maxNumAdditionalTasks,
505                                       unsigned int*   numAdditionalTasksCreated);
506
515 OPTIXAPI OptixResult optixTaskGetSerializationKey(OptixTask task, void* key, size_t* size);
516
525 OPTIXAPI OptixResult optixTaskSerializeOutput(OptixTask task, void* data, size_t* size);
526
536 OPTIXAPI OptixResult optixTaskDeserializeOutput(OptixTask      task,
537                                                  const void*    data,
538                                                  size_t        size,
539                                                  OptixTask*    additionalTasks,
540                                                  unsigned int    maxNumAdditionalTasks,
541                                                  unsigned int*   numAdditionalTasksCreated);
542
547
556 OPTIXAPI OptixResult optixProgramGroupGetStackSize(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes, OptixPipeline pipeline);
557
583 OPTIXAPI OptixResult optixProgramGroupCreate(OptixDeviceContext      context,
584                                               const OptixProgramGroupDesc* programDescriptions,
585                                               unsigned int                numProgramGroups,
586                                               const OptixProgramGroupOptions* options,
587                                               char*                      logString,
588                                               size_t*                    logStringSize,
589                                               OptixProgramGroup*        programGroups);
590
592 OPTIXAPI OptixResult optixProgramGroupDestroy(OptixProgramGroup programGroup);
593
596 OPTIXAPI OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer);
597
602
629 OPTIXAPI OptixResult optixLaunch(OptixPipeline      pipeline,
630                                   CUstream            stream,
631                                   CUdeviceptr          pipelineParams,
632                                   size_t                pipelineParamsSize,
633                                   const OptixShaderBindingTable* sbt,
634                                   unsigned int          width,
635                                   unsigned int          height,
636                                   unsigned int          depth);
637
642
648 OPTIXAPI OptixResult optixAccelComputeMemoryUsage(OptixDeviceContext      context,
649                                                     const OptixAccelBuildOptions* accelOptions,
650                                                     const OptixBuildInput*    buildInputs,
651                                                     unsigned int                numBuildInputs,
652                                                     OptixAccelBufferSizes*    bufferSizes);
653
666 OPTIXAPI OptixResult optixAccelBuild(OptixDeviceContext      context,
667                                       CUstream                stream,
668                                       const OptixAccelBuildOptions* accelOptions,
669                                       const OptixBuildInput*    buildInputs,
670                                       unsigned int                numBuildInputs,
671                                       CUdeviceptr              tempBuffer,
672                                       size_t                    tempBufferSizeInBytes,
673                                       CUdeviceptr              outputBuffer,
674                                       size_t                    outputBufferSizeInBytes,

```

```

675         OptixTraversableHandle*      outputHandle,
676         const OptixAccelEmitDesc*     emittedProperties,
677         unsigned int                  numEmittedProperties);
678
696 OPTIXAPI OptixResult optixAccelGetRelocationInfo(OptixDeviceContext context, OptixTraversableHandle
handle, OptixRelocationInfo* info);
697
709 OPTIXAPI OptixResult optixCheckRelocationCompatibility(OptixDeviceContext context, const
OptixRelocationInfo* info, int* compatible);
710
748 OPTIXAPI OptixResult optixAccelRelocate(OptixDeviceContext      context,
749         CUstream          stream,
750         const OptixRelocationInfo* info,
751         const OptixRelocateInput*  relocateInputs,
752         size_t                  numRelocateInputs,
753         CUdeviceptr          targetAccel,
754         size_t                  targetAccelSizeInBytes,
755         OptixTraversableHandle*    targetHandle);
756
774 OPTIXAPI OptixResult optixAccelCompact(OptixDeviceContext      context,
775         CUstream          stream,
776         OptixTraversableHandle inputHandle,
777         CUdeviceptr          outputBuffer,
778         size_t                  outputBufferSizeInBytes,
779         OptixTraversableHandle* outputHandle);
780
789 OPTIXAPI OptixResult optixAccelEmitProperty(OptixDeviceContext      context,
790         CUstream          stream,
791         OptixTraversableHandle handle,
792         const OptixAccelEmitDesc* emittedProperty);
793
798 OPTIXAPI OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext      onDevice,
799         CUdeviceptr          pointer,
800         OptixTraversableType traversableType,
801         OptixTraversableHandle* traversableHandle);
802
803
809 OPTIXAPI OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext
context,
810         const OptixOpacityMicromapArrayBuildInput*
buildInput,
811         OptixMicromapBufferSizes* bufferSizes);
812
835 OPTIXAPI OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext      context,
836         CUstream          stream,
837         const OptixOpacityMicromapArrayBuildInput* buildInput,
838         const OptixMicromapBuffers*                buffers);
839
855 OPTIXAPI OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext context,
856         CUdeviceptr          opacityMicromapArray,
857         OptixRelocationInfo* info);
858
885 OPTIXAPI OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext      context,
886         CUstream          stream,
887         const OptixRelocationInfo* info,
888         CUdeviceptr          targetOpacityMicromapArray,
889         size_t                  targetOpacityMicromapArraySizeInBytes);
890
891
902 OPTIXAPI OptixResult optixClusterAccelComputeMemoryUsage(OptixDeviceContext      context,
903         OptixClusterAccelBuildMode      buildMode,
904         const OptixClusterAccelBuildInput* buildInput,
905         OptixAccelBufferSizes*          bufferSizes);
906
930 OPTIXAPI OptixResult optixClusterAccelBuild(OptixDeviceContext      context,
931         CUstream          stream,

```

```

932         const OptixClusterAccelBuildModeDesc* buildModeDesc,
933         const OptixClusterAccelBuildInput*   buildInput,
934         CUdeviceptr                           argsArray,
935         CUdeviceptr                           argsCount,
936         unsigned int                          argsStrideInBytes);
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972 OPTIXAPI OptixResult optixCoopVecMatrixConvert(OptixDeviceContext context,
973         CUstream stream,
974         unsigned int numNetworks,
975         const OptixNetworkDescription* inputNetworkDescription,
976         CUdeviceptr inputNetworks,
977         size_t inputNetworkStrideInBytes,
978         const OptixNetworkDescription* outputNetworkDescription,
979         CUdeviceptr outputNetworks,
980         size_t outputNetworkStrideInBytes);
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997 OPTIXAPI OptixResult optixCoopVecMatrixComputeSize(OptixDeviceContext context,
998         unsigned int N,
999         unsigned int K,
1000         OptixCoopVecElemType elementType,
1001         OptixCoopVecMatrixLayout layout,
1002         size_t rowColumnStrideInBytes,
1003         size_t* sizeInBytes);
1004
1005
1006
1007
1008
1009
1010
1011
1012 OPTIXAPI OptixResult optixDenoiserCreate(OptixDeviceContext context,
1013         OptixDenoiserModelKind modelKind,
1014         const OptixDenoiserOptions* options,
1015         OptixDenoiser* denoiser);
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045 OPTIXAPI OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context,
1046         const void* userData,
1047         size_t userDataSizeInBytes,
1048         OptixDenoiser* denoiser);
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066 OPTIXAPI OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser denoiser,
1067         unsigned int outputWidth,
1068         unsigned int outputHeight,
1069         OptixDenoiserSizes* returnSizes);
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087 OPTIXAPI OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
1088         CUstream stream,
1089         unsigned int inputWidth,
1090         unsigned int inputHeight,
1091         CUdeviceptr denoiserState,
1092         size_t denoiserStateSizeInBytes,
1093         CUdeviceptr scratch,
1094         size_t scratchSizeInBytes);
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161 OPTIXAPI OptixResult optixDenoiserInvoke(OptixDenoiser denoiser,
1162         CUstream stream,
1163         const OptixDenoiserParams* params,
1164         CUdeviceptr denoiserState,
1165         size_t denoiserStateSizeInBytes,
1166         const OptixDenoiserGuideLayer* guideLayer,
1167         const OptixDenoiserLayer* layers,
1168         unsigned int numLayers,

```

```

1169         unsigned int      inputOffsetX,
1170         unsigned int      inputOffsetY,
1171         CUdeviceptr        scratch,
1172         size_t             scratchSizeInBytes);
1173
1197 OPTIXAPI OptixResult optixDenoiserComputeIntensity(OptixDenoiser    denoiser,
1198                                                    CUstream          stream,
1199                                                    const OptixImage2D* inputImage,
1200                                                    CUdeviceptr        outputIntensity,
1201                                                    CUdeviceptr        scratch,
1202                                                    size_t             scratchSizeInBytes);
1203
1218 OPTIXAPI OptixResult optixDenoiserComputeAverageColor(OptixDenoiser    denoiser,
1219                                                        CUstream          stream,
1220                                                        const OptixImage2D* inputImage,
1221                                                        CUdeviceptr        outputAverageColor,
1222                                                        CUdeviceptr        scratch,
1223                                                        size_t             scratchSizeInBytes);
1224
1226
1227 #include "optix_function_table.h"
1228
1229 #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

```

```

41
42 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
43 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver type float2 must be defined through other
44 // means before including optix headers.
45 #include <vector_types.h>
46 #endif
47 #include "internal/optix_micromap_impl.h"
48
49 OPTIX_MICROMAP_INLINE_FUNC void optixMicromapIndexToBaseBarycentrics(unsigned int micromapTriangleIndex,
50                                                                    unsigned int subdivisionLevel,
51                                                                    float2& baseBarycentrics0,
52                                                                    float2& baseBarycentrics1,
53                                                                    float2& baseBarycentrics2)
54 {
55     optix_impl::micro2bary(micromapTriangleIndex, subdivisionLevel, baseBarycentrics0, baseBarycentrics1,
56     baseBarycentrics2);
57 }
58
59 OPTIX_MICROMAP_INLINE_FUNC float2 optixBaseBarycentricsToMicroBarycentrics(float2 baseBarycentrics,
60                                                                    float2 microVertexBaseBarycentrics[3])
61 {
62     return optix_impl::base2micro(baseBarycentrics, microVertexBaseBarycentrics);
63 }
64
65 #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
69
70
71
72
73
74
75

```

```

76
90 inline OptixResult optixUtilComputeStackSizes(const OptixStackSizes* stackSizes,
91                                               unsigned int      maxTraceDepth,
92                                               unsigned int      maxCCDepth,
93                                               unsigned int      maxDCDepth,
94                                               unsigned int*      directCallableStackSizeFromTraversal,
95                                               unsigned int*      directCallableStackSizeFromState,
96                                               unsigned int*      continuationStackSize)
97 {
98     if(!stackSizes)
99         return OPTIX_ERROR_INVALID_VALUE;
100
101     const unsigned int cssRG = stackSizes->cssRG;
102     const unsigned int cssMS = stackSizes->cssMS;
103     const unsigned int cssCH = stackSizes->cssCH;
104     const unsigned int cssAH = stackSizes->cssAH;
105     const unsigned int cssIS = stackSizes->cssIS;
106     const unsigned int cssCC = stackSizes->cssCC;
107     const unsigned int dssDC = stackSizes->dssDC;
108
109     if(directCallableStackSizeFromTraversal)
110         *directCallableStackSizeFromTraversal = maxDCDepth * dssDC;
111     if(directCallableStackSizeFromState)
112         *directCallableStackSizeFromState = maxDCDepth * dssDC;
113
114     // upper bound on continuation stack used by call trees of continuation callables
115     unsigned int cssCCTree = maxCCDepth * cssCC;
116
117     // upper bound on continuation stack used by CH or MS programs including the call tree of
118     // continuation callables
119     unsigned int cssCHorMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
120
121     // clang-format off
122     if(continuationStackSize)
123         *continuationStackSize
124             = cssRG + cssCCTree
125             + (std::max(maxTraceDepth, 1u) - 1) * cssCHorMSPlusCCTree
126             + std::min(maxTraceDepth, 1u) * std::max(cssCHorMSPlusCCTree, cssIS + cssAH);
127     // clang-format on
128
129     return OPTIX_SUCCESS;
130 }
131
132 inline OptixResult optixUtilComputeStackSizesDCSplit(const OptixStackSizes* stackSizes,
133                                                      unsigned int      dssDCFromTraversal,
134                                                      unsigned int      dssDCFromState,
135                                                      unsigned int      maxTraceDepth,
136                                                      unsigned int      maxCCDepth,
137                                                      unsigned int      maxDCDepthFromTraversal,
138                                                      unsigned int      maxDCDepthFromState,
139                                                      unsigned int*      directCallableStackSizeFromTraversal,
140                                                      unsigned int*      directCallableStackSizeFromState,
141                                                      unsigned int*      continuationStackSize)
142 {
143     if(!stackSizes)
144         return OPTIX_ERROR_INVALID_VALUE;
145
146     const unsigned int cssRG = stackSizes->cssRG;
147     const unsigned int cssMS = stackSizes->cssMS;
148     const unsigned int cssCH = stackSizes->cssCH;
149     const unsigned int cssAH = stackSizes->cssAH;
150     const unsigned int cssIS = stackSizes->cssIS;
151     const unsigned int cssCC = stackSizes->cssCC;
152     // use dssDCFromTraversal and dssDCFromState instead of stackSizes->dssDC
153
154     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
270 inline OptixResult optixUtilComputeStackSizesSimplePathTracer(OptixProgramGroup      programGroupRG,
271                                                                OptixProgramGroup      programGroupMS1,
272                                                                const OptixProgramGroup* programGroupCH1,
273                                                                unsigned int             programGroupCH1Count,
274                                                                OptixProgramGroup      programGroupMS2,

```

```

275                                     const OptixProgramGroup* programGroupCH2,
276                                     unsigned int                programGroupCH2Count,
277                                     unsigned int*
directCallableStackSizeFromTraversal,
278                                     unsigned int* directCallableStackSizeFromState,
279                                     unsigned int* continuationStackSize,
280                                     OptixPipeline pipeline)
281 {
282     if(!programGroupCH1 && (programGroupCH1Count > 0))
283         return OPTIX_ERROR_INVALID_VALUE;
284     if(!programGroupCH2 && (programGroupCH2Count > 0))
285         return OPTIX_ERROR_INVALID_VALUE;
286
287     OptixResult result;
288
289     OptixStackSizes stackSizesRG = {};
290     result                    = optixProgramGroupGetStackSize(programGroupRG, &stackSizesRG, pipeline);
291     if(result != OPTIX_SUCCESS)
292         return result;
293
294     OptixStackSizes stackSizesMS1 = {};
295     result                    = optixProgramGroupGetStackSize(programGroupMS1, &stackSizesMS1,
pipeline);
296     if(result != OPTIX_SUCCESS)
297         return result;
298
299     OptixStackSizes stackSizesCH1 = {};
300     for(unsigned int i = 0; i < programGroupCH1Count; ++i)
301     {
302         result = optixUtilAccumulateStackSizes(programGroupCH1[i], &stackSizesCH1, pipeline);
303         if(result != OPTIX_SUCCESS)
304             return result;
305     }
306
307     OptixStackSizes stackSizesMS2 = {};
308     result                    = optixProgramGroupGetStackSize(programGroupMS2, &stackSizesMS2,
pipeline);
309     if(result != OPTIX_SUCCESS)
310         return result;
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

```

```

340
341 #ifdef __cplusplus
342 }
343 #endif
344
345 #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 - 2025 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
69 // The function table needs to be defined in exactly one translation unit. This can be
70 // achieved by including optix_function_table_definition.h in that translation unit.
71 extern OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL;
72
73 #endif
74
75 #endif

```

```

77 }
78 #endif
79
80 #ifdef _WIN32
81 #if defined(_MSC_VER)
82 // Visual Studio produces warnings suggesting strcpy and friends being replaced with _s
83 // variants. All the string lengths and allocation sizes have been calculated and should
84 // be safe, so we are disabling this warning to increase compatibility.
85 #pragma warning(push)
86 #pragma warning(disable : 4996)
87 #endif
88 static void* optixLoadWindowsDllFromName(const char* optixDllName)
89 {
90     void* handle = NULL;
91
92     // Try the bare dll name first. This picks it up in the local path, followed by
93     // standard Windows paths.
94     handle = LoadLibraryA((LPSTR)optixDllName);
95     if(handle)
96         return handle;
97 // If we don't find it in the default dll search path, try the system paths
98
99 // Get the size of the path first, then allocate
100 unsigned int size = GetSystemDirectoryA(NULL, 0);
101 if(size == 0)
102 {
103     // Couldn't get the system path size, so bail
104     return NULL;
105 }
106 size_t pathSize = size + 1 + strlen(optixDllName);
107 char* systemPath = (char*)malloc(pathSize);
108 if(systemPath == NULL)
109     return NULL;
110 if(GetSystemDirectoryA(systemPath, size) != size - 1)
111 {
112     // Something went wrong
113     free(systemPath);
114     return NULL;
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, &regKey,
154 CM_REGISTRY_SOFTWARE) != CR_SUCCESS)
155         {
156             continue;
157         }
158         const char* valueName = "OpenGLDriverName";
159         DWORD      valueSize = 0;
160         LSTATUS     ret      = RegQueryValueExA(regKey, valueName, NULL, NULL, NULL, &valueSize);
161         if(ret != ERROR_SUCCESS)
162         {
163             RegCloseKey(regKey);
164             continue;
165         }
166         char* regValue = (char*)malloc(valueSize);
167         if(regValue == NULL)
168         {
169             RegCloseKey(regKey);
170             continue;
171         }
172         ret = RegQueryValueExA(regKey, valueName, NULL, NULL, (LPBYTE)regValue, &valueSize);
173         if(ret != ERROR_SUCCESS)
174         {
175             free(regValue);
176             RegCloseKey(regKey);
177             continue;
178         }
179         // Strip the opengl driver dll name from the string then create a new string with
180         // the path and the nvoptix.dll name
181         for(int i = (int)valueSize - 1; i >= 0 && regValue[i] != '\\'; --i)
182             regValue[i] = '\\0';
183         size_t newPathSize = strlen(regValue) + strlen(optixDllName) + 1;
184         dllPath = (char*)malloc(newPathSize);
185         if(dllPath == NULL)
186         {
187             free(regValue);
188             RegCloseKey(regKey);
189             continue;
190         }
191         strcpy(dllPath, regValue);
192         strcat(dllPath, optixDllName);
193         free(regValue);
194         RegCloseKey(regKey);
195         handle = LoadLibraryA((LPCSTR)dllPath);
196         free(dllPath);
197         if(handle)
198             break;
199     }
200     free(deviceNames);
201     return handle;
202 }
203 #if defined(_MSC_VER)
204 #pragma warning(pop)
205 #endif
206 static void* optixLoadWindowsDll( )
207 {
208     return optixLoadWindowsDllFromName("nvoptix.dll");

```

```

209 }
210 #endif
211
214
224 OPTIXAPI inline OptixResult optixInitWithHandle(void** handlePtr)
225 {
226     // Make sure these functions get initialized to zero in case the DLL and function
227     // table can't be loaded
228     OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorName = 0;
229     OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorString = 0;
230
231     if(!handlePtr)
232         return OPTIX_ERROR_INVALID_VALUE;
233
234 #ifdef _WIN32
235     *handlePtr = optixLoadWindowsDll();
236     if(!*handlePtr)
237         return OPTIX_ERROR_LIBRARY_NOT_FOUND;
238
239     void* symbol = (void*)GetProcAddress((HMODULE)*handlePtr, "optixQueryFunctionTable");
240     if(!symbol)
241         return OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND;
242 #else
243     *handlePtr = dlopen("libnvoptix.so.1", RTLD_NOW);
244     if(!*handlePtr)
245         return OPTIX_ERROR_LIBRARY_NOT_FOUND;
246
247     void* symbol = dlsym(*handlePtr, "optixQueryFunctionTable");
248     if(!symbol)
249         return OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND;
250 #endif
251
252     OptixQueryFunctionTable_t* optixQueryFunctionTable = (OptixQueryFunctionTable_t*)symbol;
253
254     return optixQueryFunctionTable(OPTIX_ABI_VERSION, 0, 0, 0, &OPTIX_FUNCTION_TABLE_SYMBOL,
255     sizeof(OPTIX_FUNCTION_TABLE_SYMBOL));
256 }
257
260 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

```

```

295
296 #ifndef OPTIX_DOXYGEN_SHOULD_SKIP_THIS
297
298 // Stub functions that forward calls to the corresponding function pointer in the function table.
299
300 OPTIXAPI inline const char* optixGetErrorName(OptixResult result)
301 {
302     if(OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorName)
303         return OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorName(result);
304
305     // If the DLL and symbol table couldn't be loaded, provide a set of error strings
306     // suitable for processing errors related to the DLL loading.
307     switch(result)
308     {
309         case OPTIX_SUCCESS:
310             return "OPTIX_SUCCESS";
311         case OPTIX_ERROR_INVALID_VALUE:
312             return "OPTIX_ERROR_INVALID_VALUE";
313         case OPTIX_ERROR_UNSUPPORTED_ABI_VERSION:
314             return "OPTIX_ERROR_UNSUPPORTED_ABI_VERSION";
315         case OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH:
316             return "OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH";
317         case OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS:
318             return "OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS";
319         case OPTIX_ERROR_LIBRARY_NOT_FOUND:
320             return "OPTIX_ERROR_LIBRARY_NOT_FOUND";
321         case OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND:
322             return "OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND";
323         case OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE:
324             return "OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE";
325         default:
326             return "Unknown OptixResult code";
327     }
328 }
329
330 OPTIXAPI inline const char* optixGetErrorString(OptixResult result)
331 {
332     if(OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorString)
333         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,
371 OptixDeviceProperty property, void* value, size_t sizeInBytes)
372 {
373     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetProperty(context, property, value,
374 sizeInBytes);
375 }
376
377 OPTIXAPI inline OptixResult optixDeviceContextSetLogCallback(OptixDeviceContext context,
378 OptixLogCallback callbackFunction,
379 void* callbackData,
380 unsigned int callbackLevel)
381 {
382     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetLogCallback(context, callbackFunction,
383 callbackData, callbackLevel);
384 }
385
386 OPTIXAPI inline OptixResult optixDeviceContextSetCacheEnabled(OptixDeviceContext context, int enabled)
387 {
388     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetCacheEnabled(context, enabled);
389 }
390
391 OPTIXAPI inline OptixResult optixDeviceContextSetCacheLocation(OptixDeviceContext context, const char*
392 location)
393 {
394     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetCacheLocation(context, location);
395 }
396
397 OPTIXAPI inline OptixResult optixDeviceContextSetCacheDatabaseSizes(OptixDeviceContext context, size_t
398 lowWaterMark, size_t highWaterMark)
399 {
400     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetCacheDatabaseSizes(context, lowWaterMark,
401 highWaterMark);
402 }
403
404 OPTIXAPI inline OptixResult optixDeviceContextGetCacheEnabled(OptixDeviceContext context, int* enabled)
405 {
406     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetCacheEnabled(context, enabled);
407 }
408
409 OPTIXAPI inline OptixResult optixDeviceContextGetCacheLocation(OptixDeviceContext context, char*
410 location, size_t locationSize)
411 {
412     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetCacheLocation(context, location,
413 locationSize);
414 }
415
416 OPTIXAPI inline OptixResult optixDeviceContextGetCacheDatabaseSizes(OptixDeviceContext context, size_t*
417 lowWaterMark, size_t* highWaterMark)
418 {
419     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetCacheDatabaseSizes(context, lowWaterMark,
420 highWaterMark);
421 }
422
423 OPTIXAPI inline OptixResult optixModuleCreate(OptixDeviceContext context,
424 const OptixModuleCompileOptions* moduleCompileOptions,
425 const OptixPipelineCompileOptions* pipelineCompileOptions,
426 const char* input,
427 size_t inputSize,

```

```

418         char*                logString,
419         size_t*              logStringSize,
420         OptixModule*         module)
421 {
422     return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleCreate(context, moduleCompileOptions,
423 pipelineCompileOptions, input,
424                                     inputSize, logString, logStringSize, module);
425 }
426 OPTIXAPI inline OptixResult optixModuleCreateWithTasks(OptixDeviceContext context,
427 const OptixModuleCompileOptions*
428 moduleCompileOptions,
429 const OptixPipelineCompileOptions*
430 pipelineCompileOptions,
431 const char*                input,
432 size_t                    inputSize,
433 char*                    logString,
434 size_t*                  logStringSize,
435 OptixModule*              module,
436 OptixTask*                firstTask)
437 {
438     return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleCreateWithTasks(context, moduleCompileOptions,
439 pipelineCompileOptions, input,
440                                     inputSize, logString, logStringSize,
441 module, firstTask);
442 }
443
444 OPTIXAPI inline OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState*
445 state)
446 {
447     return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleGetCompilationState(module, state);
448 }
449
450 OPTIXAPI inline OptixResult optixModuleCancelCreation(OptixModule module, OptixCreationFlags flags)
451 {
452     return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleCancelCreation(module, flags);
453 }
454
455 OPTIXAPI inline OptixResult optixDeviceContextCancelCreations(OptixDeviceContext context,
456 OptixCreationFlags flags)
457 {
458     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextCancelCreations(context, flags);
459 }
460
461 OPTIXAPI inline OptixResult optixModuleDestroy(OptixModule module)
462 {
463     return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleDestroy(module);
464 }
465
466 OPTIXAPI inline OptixResult optixBuiltinISModuleGet(OptixDeviceContext context,
467 const OptixModuleCompileOptions* moduleCompileOptions,
468 const OptixPipelineCompileOptions* pipelineCompileOptions,
469 const OptixBuiltinISOOptions* builtinISOOptions,
470 OptixModule* builtinModule)
471 {
472     return OPTIX_FUNCTION_TABLE_SYMBOL.optixBuiltinISModuleGet(context, moduleCompileOptions,
473 pipelineCompileOptions,
474                                     builtinISOOptions, builtinModule);
475 }
476
477 OPTIXAPI inline OptixResult optixTaskExecute(OptixTask task,
478 OptixTask* additionalTasks,
479 unsigned int maxNumAdditionalTasks,
480 unsigned int* numAdditionalTasksCreated)
481 {
482     return OPTIX_FUNCTION_TABLE_SYMBOL.optixTaskExecute(task, additionalTasks, maxNumAdditionalTasks,

```

```

numAdditionalTasksCreated);
477 }
478
479 OPTIXAPI inline OptixResult optixTaskGetSerializationKey(OptixTask task, void* key, size_t* size)
480 {
481     return OPTIX_FUNCTION_TABLE_SYMBOL.optixTaskGetSerializationKey(task, key, size);
482 }
483
484 OPTIXAPI inline OptixResult optixTaskSerializeOutput(OptixTask task, void* data, size_t* size)
485 {
486     return OPTIX_FUNCTION_TABLE_SYMBOL.optixTaskSerializeOutput(task, data, size);
487 }
488
489 OPTIXAPI inline OptixResult optixTaskDeserializeOutput(OptixTask task,
490                                                         const void* data,
491                                                         size_t size,
492                                                         OptixTask* additionalTasks,
493                                                         unsigned int maxNumAdditionalTasks,
494                                                         unsigned int* numAdditionalTasksCreated)
495 {
496     return OPTIX_FUNCTION_TABLE_SYMBOL.optixTaskDeserializeOutput(task, data, size, additionalTasks,
497                                                                     maxNumAdditionalTasks,
498                                                                     numAdditionalTasksCreated);
499 }
500
501 OPTIXAPI inline OptixResult optixProgramGroupCreate(OptixDeviceContext context,
502                                                         const OptixProgramGroupDesc* programDescriptions,
503                                                         unsigned int numProgramGroups,
504                                                         const OptixProgramGroupOptions* options,
505                                                         char* logString,
506                                                         size_t* logStringSize,
507                                                         OptixProgramGroup* programGroups)
508 {
509     return OPTIX_FUNCTION_TABLE_SYMBOL.optixProgramGroupCreate(context, programDescriptions,
510                                                                     numProgramGroups, options,
511                                                                     logString, logStringSize, programGroups);
512 }
513
514 OPTIXAPI inline OptixResult optixProgramGroupDestroy(OptixProgramGroup programGroup)
515 {
516     return OPTIX_FUNCTION_TABLE_SYMBOL.optixProgramGroupDestroy(programGroup);
517 }
518
519 OPTIXAPI inline OptixResult optixProgramGroupGetStackSize(OptixProgramGroup programGroup,
520 OptixStackSizes* stackSizes, OptixPipeline pipeline)
521 {
522     return OPTIX_FUNCTION_TABLE_SYMBOL.optixProgramGroupGetStackSize(programGroup, stackSizes, pipeline);
523 }
524
525 OPTIXAPI inline OptixResult optixPipelineCreate(OptixDeviceContext context,
526                                                         const OptixPipelineCompileOptions* pipelineCompileOptions,
527                                                         const OptixPipelineLinkOptions* pipelineLinkOptions,
528                                                         const OptixProgramGroup* programGroups,
529                                                         unsigned int numProgramGroups,
530                                                         char* logString,
531                                                         size_t* logStringSize,
532                                                         OptixPipeline* pipeline)
533 {
534     return OPTIX_FUNCTION_TABLE_SYMBOL.optixPipelineCreate(context, pipelineCompileOptions,
535                                                                     pipelineLinkOptions, programGroups,
536                                                                     numProgramGroups, logString, logStringSize,
537                                                                     pipeline);
538 }
539
540 OPTIXAPI inline OptixResult optixPipelineDestroy(OptixPipeline pipeline)
541 {
542     return OPTIX_FUNCTION_TABLE_SYMBOL.optixPipelineDestroy(pipeline);
543 }

```

```

538 }
539
540 OPTIXAPI inline OptixResult optixPipelineSetStackSizeFromCallDepths(OptixPipeline pipeline,
541                                                                    unsigned int maxTraceDepth,
542                                                                    unsigned int maxContinuationCallableDepth,
543                                                                    unsigned int
544                                                                    maxDirectCallableDepthFromState,
545                                                                    unsigned int
546                                                                    maxDirectCallableDepthFromTraversal,
547                                                                    unsigned int maxTraversableGraphDepth)
548 {
549     return OPTIX_FUNCTION_TABLE_SYMBOL.optixPipelineSetStackSize(pipeline, maxTraceDepth,
550                                                                    maxContinuationCallableDepth,
551                                                                    maxDirectCallableDepthFromState,
552                                                                    maxDirectCallableDepthFromTraversal, maxTraversableGraphDepth);
553 }
554
555 OPTIXAPI inline OptixResult optixPipelineSetStackSize(OptixPipeline pipeline,
556                                                                    unsigned int directCallableStackSizeFromTraversal,
557                                                                    unsigned int directCallableStackSizeFromState,
558                                                                    unsigned int continuationStackSize,
559                                                                    unsigned int maxTraversableGraphDepth)
560 {
561     return OPTIX_FUNCTION_TABLE_SYMBOL.optixPipelineSetStackSize(pipeline,
562                                                                    directCallableStackSizeFromTraversal,
563                                                                    directCallableStackSizeFromState,
564                                                                    continuationStackSize,
565                                                                    maxTraversableGraphDepth);
566 }
567
568 OPTIXAPI inline OptixResult optixPipelineSymbolMemcpyAsync(OptixPipeline pipeline,
569                                                                    const char* name,
570                                                                    void* mem,
571                                                                    size_t sizeInBytes,
572                                                                    size_t offsetInBytes,
573                                                                    OptixPipelineSymbolMemcpyKind kind,
574                                                                    CUstream stream)
575 {
576     return OPTIX_FUNCTION_TABLE_SYMBOL.optixPipelineSymbolMemcpyAsync(pipeline, name, mem, sizeInBytes,
577                                                                    offsetInBytes, kind, stream);
578 }
579
580 OPTIXAPI inline OptixResult optixAccelComputeMemoryUsage(OptixDeviceContext context,
581                                                                    const OptixAccelBuildOptions* accelOptions,
582                                                                    const OptixBuildInput* buildInputs,
583                                                                    unsigned int numBuildInputs,
584                                                                    OptixAccelBufferSizes* bufferSizes)
585 {
586     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelComputeMemoryUsage(context, accelOptions, buildInputs,
587                                                                    numBuildInputs, bufferSizes);
588 }
589
590 OPTIXAPI inline OptixResult optixAccelBuild(OptixDeviceContext context,
591                                                                    CUstream stream,
592                                                                    const OptixAccelBuildOptions* accelOptions,
593                                                                    const OptixBuildInput* buildInputs,
594                                                                    unsigned int numBuildInputs,
595                                                                    CUdeviceptr tempBuffer,
596                                                                    size_t tempBufferSizeInBytes,
597                                                                    CUdeviceptr outputBuffer,
598                                                                    size_t outputBufferSizeInBytes,
599                                                                    OptixTraversableHandle* outputHandle,
600                                                                    const OptixAccelEmitDesc* emittedProperties,
601                                                                    unsigned int numEmittedProperties)
602 {
603     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelBuild(context, stream, accelOptions, buildInputs,

```

```

numBuildInputs, tempBuffer,
597                                     tempBufferSizeInBytes, outputBuffer,
outputBufferSizeInBytes,
598                                     outputHandle, emittedProperties, numEmittedProperties);
599 }
600
601
602 OPTIXAPI inline OptixResult optixAccelGetRelocationInfo(OptixDeviceContext context,
OptixTraversableHandle handle, OptixRelocationInfo* info)
603 {
604     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelGetRelocationInfo(context, handle, info);
605 }
606
607
608 OPTIXAPI inline OptixResult optixCheckRelocationCompatibility(OptixDeviceContext context, const
OptixRelocationInfo* info, int* compatible)
609 {
610     return OPTIX_FUNCTION_TABLE_SYMBOL.optixCheckRelocationCompatibility(context, info, compatible);
611 }
612
613 OPTIXAPI inline OptixResult optixAccelRelocate(OptixDeviceContext context,
614                                     CUstream stream,
615                                     const OptixRelocationInfo* info,
616                                     const OptixRelocateInput* relocateInputs,
617                                     size_t numRelocateInputs,
618                                     CUdeviceptr targetAccel,
619                                     size_t targetAccelSizeInBytes,
620                                     OptixTraversableHandle* targetHandle)
621 {
622     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelRelocate(context, stream, info, relocateInputs,
numRelocateInputs,
623                                     targetAccel, targetAccelSizeInBytes, targetHandle);
624 }
625
626 OPTIXAPI inline OptixResult optixAccelCompact(OptixDeviceContext context,
627                                     CUstream stream,
628                                     OptixTraversableHandle inputHandle,
629                                     CUdeviceptr outputBuffer,
630                                     size_t outputBufferSizeInBytes,
631                                     OptixTraversableHandle* outputHandle)
632 {
633     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelCompact(context, stream, inputHandle, outputBuffer,
outputBufferSizeInBytes, outputHandle);
634 }
635 }
636
637 OPTIXAPI inline OptixResult optixAccelEmitProperty(OptixDeviceContext context,
638                                     CUstream stream,
639                                     OptixTraversableHandle handle,
640                                     const OptixAccelEmitDesc* emittedProperty)
641 {
642     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelEmitProperty(context, stream, handle, emittedProperty);
643 }
644
645 OPTIXAPI inline OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext onDevice,
646                                     CUdeviceptr pointer,
647                                     OptixTraversableType traversableType,
648                                     OptixTraversableHandle* traversableHandle)
649 {
650     return OPTIX_FUNCTION_TABLE_SYMBOL.optixConvertPointerToTraversableHandle(onDevice, pointer,
traversableType, traversableHandle);
651 }
652
653 OPTIXAPI inline OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext context,
654                                     const
OptixOpacityMicromapArrayBuildInput* buildInput,
655                                     OptixMicromapBufferSizes* bufferSizes)
656 {

```

```

657     return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayComputeMemoryUsage(context, buildInput,
658 bufferSizes);
659 }
660 OPTIXAPI inline OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext
661 context,
662                                     CUstream stream,
663                                     const OptixOpacityMicromapArrayBuildInput*
664 buildInput,
665                                     const OptixMicromapBuffers* buffers)
666 {
667     return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayBuild(context, stream, buildInput,
668 buffers);
669 }
670 OPTIXAPI inline OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext context,
671 CUdeviceptr opacityMicromapArray,
672                                     OptixRelocationInfo* info)
673 {
674     return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayGetRelocationInfo(context,
675 opacityMicromapArray, info);
676 }
677 OPTIXAPI inline OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext context,
678 CUstream stream,
679                                     const OptixRelocationInfo* info,
680 CUdeviceptr
681 targetOpacityMicromapArray,
682                                     size_t targetOpacityMicromapArraySizeInBytes)
683 {
684     return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayRelocate(context, stream, info,
685 targetOpacityMicromapArray,
686                                     targetOpacityMicromapArraySizeInBytes);
687 }
688 OPTIXAPI inline OptixResult optixClusterAccelComputeMemoryUsage(OptixDeviceContext
689 context,
690                                     OptixClusterAccelBuildMode buildMode,
691                                     const OptixClusterAccelBuildInput* buildInput,
692                                     OptixAccelBufferSizes* bufferSizes)
693 {
694     return OPTIX_FUNCTION_TABLE_SYMBOL.optixClusterAccelComputeMemoryUsage(context, buildMode,
695 buildInput, bufferSizes);
696 }
697 OPTIXAPI inline OptixResult optixClusterAccelBuild(OptixDeviceContext context,
698 CUstream stream,
699                                     const OptixClusterAccelBuildModeDesc* buildModeDesc,
700                                     const OptixClusterAccelBuildInput* buildInput,
701                                     CUdeviceptr argsArray,
702                                     CUdeviceptr argsCount,
703                                     unsigned int argsStrideInBytes)
704 {
705     return OPTIX_FUNCTION_TABLE_SYMBOL.optixClusterAccelBuild(context, stream, buildModeDesc,
706 buildInput, argsArray,
707                                     argsCount, argsStrideInBytes);
708 }
709 OPTIXAPI inline OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void*
710 sbtRecordHeaderHostPointer)
711 {
712     return OPTIX_FUNCTION_TABLE_SYMBOL.optixSbtRecordPackHeader(programGroup,
713 sbtRecordHeaderHostPointer);
714 }
715 OPTIXAPI inline OptixResult optixLaunch(OptixPipeline pipeline,

```

```

712         CUstream          stream,
713         CUdeviceptr        pipelineParams,
714         size_t             pipelineParamsSize,
715         const OptixShaderBindingTable* sbt,
716         unsigned int        width,
717         unsigned int        height,
718         unsigned int        depth)
719 {
720     return OPTIX_FUNCTION_TABLE_SYMBOL.optixLaunch(pipeline, stream, pipelineParams, pipelineParamsSize,
721     sbt, width, height, depth);
722 }
723 OPTIXAPI inline OptixResult optixCoopVecMatrixConvert(OptixDeviceContext context,
724     CUstream          stream,
725     unsigned int      numNetworks,
726     const OptixNetworkDescription* inputNetworkDescription,
727     CUdeviceptr        inputNetworks,
728     size_t            inputNetworkStrideInBytes,
729     const OptixNetworkDescription* outputNetworkDescription,
730     CUdeviceptr        outputNetworks,
731     size_t            outputNetworkStrideInBytes)
732 {
733     return OPTIX_FUNCTION_TABLE_SYMBOL.optixCoopVecMatrixConvert(context, stream, numNetworks,
734     inputNetworkDescription,
735     inputNetworks, inputNetworkStrideInBytes,
736     outputNetworkDescription,
737     outputNetworks, outputNetworkStrideInBytes);
738 }
739 OPTIXAPI inline OptixResult optixCoopVecMatrixComputeSize(OptixDeviceContext context,
740     unsigned int      N,
741     unsigned int      K,
742     OptixCoopVecElemType elementType,
743     OptixCoopVecMatrixLayout layout,
744     size_t            rowColumnStrideInBytes,
745     size_t*           sizeInBytes)
746 {
747     return OPTIX_FUNCTION_TABLE_SYMBOL.optixCoopVecMatrixComputeSize(context, N, K, elementType, layout,
748     rowColumnStrideInBytes, sizeInBytes);
749 }
750 OPTIXAPI inline OptixResult optixDenoiserCreate(OptixDeviceContext context,
751     OptixDenoiserModelKind modelKind,
752     const OptixDenoiserOptions* options,
753     OptixDenoiser*          returnHandle)
754 {
755     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserCreate(context, modelKind, options, returnHandle);
756 }
757 OPTIXAPI inline OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context,
758     const void*      data,
759     size_t           dataSizeInBytes,
760     OptixDenoiser*   returnHandle)
761 {
762     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserCreateWithUserModel(context, data, dataSizeInBytes,
763     returnHandle);
764 }
765 OPTIXAPI inline OptixResult optixDenoiserDestroy(OptixDenoiser handle)
766 {
767     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserDestroy(handle);
768 }
769 OPTIXAPI inline OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser handle,
770     unsigned int      maximumInputWidth,
771     unsigned int      maximumInputHeight,
772     OptixDenoiserSizes* returnSizes)

```

```

775 {
776     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserComputeMemoryResources(handle, maximumInputWidth,
maximumInputHeight, returnSizes);
777 }
778
779 OPTIXAPI inline OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
780         CUstream stream,
781         unsigned int inputWidth,
782         unsigned int inputHeight,
783         CUdeviceptr denoiserState,
784         size_t denoiserStateSizeInBytes,
785         CUdeviceptr scratch,
786         size_t scratchSizeInBytes)
787 {
788     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserSetup(denoiser, stream, inputWidth, inputHeight,
denoiserState,
789         denoiserStateSizeInBytes, scratch,
scratchSizeInBytes);
790 }
791
792 OPTIXAPI inline OptixResult optixDenoiserInvoke(OptixDenoiser handle,
793         CUstream stream,
794         const OptixDenoiserParams* params,
795         CUdeviceptr denoiserData,
796         size_t denoiserDataSize,
797         const OptixDenoiserGuideLayer* guideLayer,
798         const OptixDenoiserLayer* layers,
799         unsigned int numLayers,
800         unsigned int inputOffsetX,
801         unsigned int inputOffsetY,
802         CUdeviceptr scratch,
803         size_t scratchSizeInBytes)
804 {
805     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserInvoke(handle, stream, params, denoiserData,
denoiserDataSize,
806         guideLayer, layers, numLayers, inputOffsetX,
inputOffsetY,
807         scratch, scratchSizeInBytes);
808 }
809
810 OPTIXAPI inline OptixResult optixDenoiserComputeIntensity(OptixDenoiser handle,
811         CUstream stream,
812         const OptixImage2D* inputImage,
813         CUdeviceptr outputIntensity,
814         CUdeviceptr scratch,
815         size_t scratchSizeInBytes)
816 {
817     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserComputeIntensity(handle, stream, inputImage,
outputIntensity,
818         scratch, scratchSizeInBytes);
819 }
820
821 OPTIXAPI inline OptixResult optixDenoiserComputeAverageColor(OptixDenoiser handle,
822         CUstream stream,
823         const OptixImage2D* inputImage,
824         CUdeviceptr outputAverageColor,
825         CUdeviceptr scratch,
826         size_t scratchSizeInBytes)
827 {
828     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserComputeAverageColor(handle, stream, inputImage,
outputAverageColor,
829         scratch, scratchSizeInBytes);
830 }
831
832 #endif // OPTIX_DOXYGEN_SHOULD_SKIP_THIS
833
834 #endif // OPTIX_OPTIX_STUBS_H

```

8.25 optix_types.h File Reference

Classes

- struct OptixDeviceContextOptions
- struct OptixOpacityMicromapUsageCount
- struct OptixBuildInputOpacityMicromap
- struct OptixRelocateInputOpacityMicromap
- 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 OptixClusterAccelBuildModeDescImplicitDest
- struct OptixClusterAccelBuildModeDescExplicitDest
- struct OptixClusterAccelBuildModeDescGetSize
- struct OptixClusterAccelBuildInputTriangles
- struct OptixClusterAccelBuildInputGrids
- struct OptixClusterAccelBuildInputClusters
- struct OptixClusterAccelPrimitiveInfo
- struct OptixClusterAccelBuildInputTrianglesArgs
- struct OptixClusterAccelBuildInputGridsArgs
- struct OptixClusterAccelBuildInputTemplatesArgs
- struct OptixClusterAccelBuildInputClustersArgs
- struct OptixClusterAccelBuildInput
- struct OptixClusterAccelBuildModeDesc
- struct OptixImage2D
- struct OptixDenoiserOptions
- struct OptixDenoiserGuideLayer
- struct OptixDenoiserLayer
- struct OptixDenoiserParams

- struct OptixDenoiserSizes
- struct OptixTraverseData
- struct OptixModuleCompileBoundValueEntry
- struct OptixPayloadType
- struct OptixModuleCompileOptions
- struct OptixBuiltinISOOptions
- struct OptixProgramGroupSingleModule
- struct OptixProgramGroupHitgroup
- struct OptixProgramGroupCallables
- struct OptixProgramGroupDesc
- struct OptixProgramGroupOptions
- struct OptixPipelineCompileOptions
- struct OptixPipelineLinkOptions
- struct OptixShaderBindingTable
- struct OptixStackSizes
- struct OptixCoopVecMatrixDescription
- struct OptixNetworkDescription

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_PREDEFINED_INDEX_CLUSTER_SKIP_OPACITY_MICROMAP (-5)
- #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
- #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12

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 OptixPipelineSymbolMemcpyKind OptixPipelineSymbolMemcpyKind
- typedef enum OptixDevicePropertyShaderExecutionReorderingFlags OptixDevicePropertyShaderExecutionReorderingFlags
- typedef enum OptixDevicePropertyClusterAccelFlags OptixDevicePropertyClusterAccelFlags
- typedef enum OptixGeometryFlags OptixGeometryFlags
- typedef enum OptixHitKind OptixHitKind
- typedef enum OptixIndicesFormat OptixIndicesFormat
- typedef enum OptixVertexFormat OptixVertexFormat
- typedef enum OptixTransformFormat OptixTransformFormat
- typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat
- typedef enum OptixOpacityMicromapArrayIndexingMode OptixOpacityMicromapArrayIndexingMode
- typedef struct OptixOpacityMicromapUsageCount OptixOpacityMicromapUsageCount
- typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap
- typedef struct OptixRelocateInputOpacityMicromap OptixRelocateInputOpacityMicromap
- 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 OptixClusterAccelBuildFlags OptixClusterAccelBuildFlags
- typedef enum OptixClusterAccelClusterFlags OptixClusterAccelClusterFlags
- typedef enum OptixClusterAccelPrimitiveFlags OptixClusterAccelPrimitiveFlags
- typedef enum OptixClusterAccelBuildType OptixClusterAccelBuildType
- typedef enum OptixClusterAccelBuildMode OptixClusterAccelBuildMode
- typedef enum OptixClusterAccelIndicesFormat OptixClusterAccelIndicesFormat
- typedef struct OptixClusterAccelBuildModeDescImplicitDest
OptixClusterAccelBuildModeDescImplicitDest
- typedef struct OptixClusterAccelBuildModeDescExplicitDest
OptixClusterAccelBuildModeDescExplicitDest
- typedef struct OptixClusterAccelBuildModeDescGetSize
OptixClusterAccelBuildModeDescGetSize
- typedef struct OptixClusterAccelBuildInputTriangles OptixClusterAccelBuildInputTriangles
- typedef struct OptixClusterAccelBuildInputGrids OptixClusterAccelBuildInputGrids
- typedef struct OptixClusterAccelBuildInputClusters OptixClusterAccelBuildInputClusters
- typedef struct OptixClusterAccelPrimitiveInfo OptixClusterAccelPrimitiveInfo
- typedef enum OptixClusterIDValues OptixClusterIDValues
- typedef struct OptixClusterAccelBuildInputTrianglesArgs
OptixClusterAccelBuildInputTrianglesArgs
- typedef struct OptixClusterAccelBuildInputGridsArgs OptixClusterAccelBuildInputGridsArgs
- typedef struct OptixClusterAccelBuildInputTemplatesArgs
OptixClusterAccelBuildInputTemplatesArgs
- typedef struct OptixClusterAccelBuildInputClustersArgs
OptixClusterAccelBuildInputClustersArgs
- typedef struct OptixClusterAccelBuildInput OptixClusterAccelBuildInput
- typedef struct OptixClusterAccelBuildModeDesc OptixClusterAccelBuildModeDesc
- 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 struct OptixTraverseData OptixTraverseData
- typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags
- typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
- typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
- typedef enum OptixModuleCompileState OptixModuleCompileState
- typedef enum OptixCreationFlags OptixCreationFlags
- typedef struct OptixModuleCompileBoundValueEntry OptixModuleCompileBoundValueEntry
- typedef enum OptixPayloadTypeID OptixPayloadTypeID
- typedef enum OptixPayloadSemantics OptixPayloadSemantics
- typedef struct OptixPayloadType OptixPayloadType
- typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
- typedef struct OptixBuiltinISOOptions OptixBuiltinISOOptions
- 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 OptixDevicePropertyCoopVecFlags OptixDevicePropertyCoopVecFlags
- typedef enum OptixCoopVecElemType OptixCoopVecElemType
- typedef enum OptixCoopVecMatrixLayout OptixCoopVecMatrixLayout
- typedef struct OptixCoopVecMatrixDescription OptixCoopVecMatrixDescription
- typedef struct OptixNetworkDescription OptixNetworkDescription
- typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
- typedef OptixResult() OptixQueryFunctionTable_t(int abiId, unsigned int numOptions, OptixQueryFunctionTableOptions *, const void **, void *functionTable, size_t sizeOfTable)

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_CREATION_CANCELED = 7290 ,
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_INVALID_POINTER = 7808 ,
OPTIX_ERROR_SYMBOL_NOT_FOUND = 7809 ,
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 ,
    OPTIX_DEVICE_PROPERTY_COOP_VEC = 0x200B ,
    OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL = 0x2020 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_VERTICES = 0x2021 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_TRIANGLES = 0x2022 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_STRUCTURED_GRID_RESOLUTION = 0x2023 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_SBT_INDEX = 0x2024 ,
    OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTERS_PER_GAS = 0x2025 }

```

- enum OptixDeviceContextValidationMode {
OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0 ,
OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF }
- enum OptixPipelineSymbolMemcpyKind {
OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_FROM_DEVICE = 0x21A0 ,
OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_FROM_HOST = 0x21A1 ,
OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_TO_DEVICE = 0x21A2 ,
OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_TO_HOST = 0x21A3 }
- enum OptixDevicePropertyShaderExecutionReorderingFlags {
OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_NONE = 0 ,
OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD = 1
<< 0 }
- enum OptixDevicePropertyClusterAccelFlags {
OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_NONE = 0 ,
OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_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 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 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_ROUND_QUADRATIC_BSPLINE_ROCAPS = 0x2508 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE_ROCAPS = 0x2509 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM_ROCAPS = 0x250A ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER_ROCAPS = 0x250B ,
OPTIX_PRIMITIVE_TYPE_TRIANGLE = 0x2531 }

```

- 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_ROUND_QUADRATIC_BSPLINE_ROCAPS = 1 << 8 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE_ROCAPS = 1 << 9 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM_ROCAPS = 1 << 10 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER_ROCAPS = 1 << 11 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE = 1 << 31 }

```
- 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 OptixClusterAccelBuildFlags {  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_FLAG\_NONE = 0 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_FLAG\_PREFER\_FAST\_TRACE = 1 << 0 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_FLAG\_PREFER\_FAST\_BUILD = 1 << 1 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_FLAG\_ALLOW\_OPACITY\_MICROMAPS = 1 << 2 }
  - enum OptixClusterAccelClusterFlags {  
OPTIX\_CLUSTER\_ACCEL\_CLUSTER\_FLAG\_NONE = 0 ,  
OPTIX\_CLUSTER\_ACCEL\_CLUSTER\_FLAG\_ALLOW\_DISABLE\_OPACITY\_MICROMAPS = 1  
<< 0 }
  - enum OptixClusterAccelPrimitiveFlags {  
OPTIX\_CLUSTER\_ACCEL\_PRIMITIVE\_FLAG\_NONE = 0 ,  
OPTIX\_CLUSTER\_ACCEL\_PRIMITIVE\_FLAG\_DISABLE\_TRIANGLE\_FACE\_CULLING = 1 <<  
0 ,  
OPTIX\_CLUSTER\_ACCEL\_PRIMITIVE\_FLAG\_REQUIRE\_SINGLE\_ANYHIT\_CALL = 1 << 1 ,  
OPTIX\_CLUSTER\_ACCEL\_PRIMITIVE\_FLAG\_DISABLE\_ANYHIT = 1 << 2 }
  - enum OptixClusterAccelBuildType {  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_GASES\_FROM\_CLUSTERS = 0x2545 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_CLUSTERS\_FROM\_TRIANGLES = 0x2546 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_TEMPLATES\_FROM\_TRIANGLES = 0x2547 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_CLUSTERS\_FROM\_TEMPLATES = 0x2548 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_TYPE\_TEMPLATES\_FROM\_GRIDS = 0x2549 }
  - enum OptixClusterAccelBuildMode {  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_MODE\_IMPLICIT\_DESTINATIONS = 0 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_MODE\_EXPLICIT\_DESTINATIONS = 1 ,  
OPTIX\_CLUSTER\_ACCEL\_BUILD\_MODE\_GET\_SIZES = 2 }
  - enum OptixClusterAccelIndicesFormat {  
OPTIX\_CLUSTER\_ACCEL\_INDICES\_FORMAT\_8BIT = 1 ,  
OPTIX\_CLUSTER\_ACCEL\_INDICES\_FORMAT\_16BIT = 2 ,  
OPTIX\_CLUSTER\_ACCEL\_INDICES\_FORMAT\_32BIT = 4 }
  - enum OptixClusterIDValues { OPTIX\_CLUSTER\_ID\_INVALID = 0xFFFFFFFFFu }
  - 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_AOV = 0x2324 ,
OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326 ,
OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327 ,
OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328 ,
OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322 ,
OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323 ,
OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325 }

```
 - **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 OptixCreationFlags {
    OPTIX_CREATION_FLAG_NONE = 0 ,
    OPTIX_CREATION_FLAG_BLOCK_UNTIL_EFFECTIVE = 1 << 0 }

• 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 OptixDevicePropertyCoopVecFlags {
    OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_NONE = 0 ,
    OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_STANDARD = 1 << 0 }
• enum OptixCoopVecElemType {
    OPTIX_COOP_VEC_ELEM_TYPE_UNKNOWN = 0x2A00 ,
    OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16 = 0x2A01 ,
    OPTIX_COOP_VEC_ELEM_TYPE_FLOAT32 = 0x2A03 ,
    OPTIX_COOP_VEC_ELEM_TYPE_UINT8 = 0x2A04 ,
    OPTIX_COOP_VEC_ELEM_TYPE_INT8 = 0x2A05 ,
    OPTIX_COOP_VEC_ELEM_TYPE_UINT32 = 0x2A08 ,
    OPTIX_COOP_VEC_ELEM_TYPE_INT32 = 0x2A09 ,
    OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E4M3 = 0x2A0A ,
    OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E5M2 = 0x2A0B }
• enum OptixCoopVecMatrixLayout {
    OPTIX_COOP_VEC_MATRIX_LAYOUT_ROW_MAJOR = 0x2A40 ,
    OPTIX_COOP_VEC_MATRIX_LAYOUT_COLUMN_MAJOR = 0x2A41 ,
    OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL = 0x2A42 ,
    OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL = 0x2A43 }
• 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.

8.26 optix_types.h

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

```

1
2 /*
3  * SPDX-FileCopyrightText: Copyright (c) 2019 - 2025 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 #ifndef OPTIX_OPTIX_TYPES_H
16 #define OPTIX_OPTIX_TYPES_H
17
18 #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 // This typedef should match the one in cuda.h in order to avoid compilation errors.
37 #if defined(_WIN64) || defined(__LP64__)
38 typedef unsigned long long CUdeviceptr;
39 #else
40 typedef unsigned int CUdeviceptr;
41 #endif
42
43
44 typedef struct OptixDeviceContext_t* OptixDeviceContext;
45
46 typedef struct OptixModule_t* OptixModule;
47
48
49 typedef struct OptixProgramGroup_t* OptixProgramGroup;
50
51
52 typedef struct OptixPipeline_t* OptixPipeline;
53
54
55 typedef struct OptixDenoiser_t* OptixDenoiser;
56
57
58 typedef struct OptixTask_t* OptixTask;
59
60
61 typedef unsigned long long OptixTraversableHandle;
62
63
64 typedef unsigned int OptixVisibilityMask;
65
66
67 #define OPTIX_SBT_RECORD_HEADER_SIZE ((size_t)32)
68
69 #define OPTIX_SBT_RECORD_ALIGNMENT 16ull
70
71 #define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128ull
72
73 #define OPTIX_INSTANCE_BYTE_ALIGNMENT 16ull
74
75 #define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8ull
76
77 #define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16ull
78
79 #define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
80
81 #define OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
82
83 #define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
84
85 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
86
87 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
88
89 #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT (0)
90 #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE (1)
91 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT (2)
92 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE (3)
93
94 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT (-1)
95 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE (-2)
96 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT (-3)
97 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE (-4)
98 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_CLUSTER_SKIP_OPACITY_MICROMAP (-5)
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124

```

```

126 #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
127
129 #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
130
138 typedef enum OptixResult
139 {
140     OPTIX_SUCCESS = 0,
141     OPTIX_ERROR_INVALID_VALUE = 7001,
142     OPTIX_ERROR_HOST_OUT_OF_MEMORY = 7002,
143     OPTIX_ERROR_INVALID_OPERATION = 7003,
144     OPTIX_ERROR_FILE_IO_ERROR = 7004,
145     OPTIX_ERROR_INVALID_FILE_FORMAT = 7005,
146     OPTIX_ERROR_DISK_CACHE_INVALID_PATH = 7010,
147     OPTIX_ERROR_DISK_CACHE_PERMISSION_ERROR = 7011,
148     OPTIX_ERROR_DISK_CACHE_DATABASE_ERROR = 7012,
149     OPTIX_ERROR_DISK_CACHE_INVALID_DATA = 7013,
150     OPTIX_ERROR_LAUNCH_FAILURE = 7050,
151     OPTIX_ERROR_INVALID_DEVICE_CONTEXT = 7051,
152     OPTIX_ERROR_CUDA_NOT_INITIALIZED = 7052,
153     OPTIX_ERROR_VALIDATION_FAILURE = 7053,
154     OPTIX_ERROR_INVALID_INPUT = 7200,
155     OPTIX_ERROR_INVALID_LAUNCH_PARAMETER = 7201,
156     OPTIX_ERROR_INVALID_PAYLOAD_ACCESS = 7202,
157     OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS = 7203,
158     OPTIX_ERROR_INVALID_FUNCTION_USE = 7204,
159     OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS = 7205,
160     OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250,
161     OPTIX_ERROR_PIPELINE_LINK_ERROR = 7251,
162     OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE = 7270,
163     OPTIX_ERROR_CREATION_CANCELED = 7290,
164     OPTIX_ERROR_INTERNAL_COMPILER_ERROR = 7299,
165     OPTIX_ERROR_DENOISER_MODEL_NOT_SET = 7300,
166     OPTIX_ERROR_DENOISER_NOT_INITIALIZED = 7301,
167     OPTIX_ERROR_NOT_COMPATIBLE = 7400,
168     OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH = 7500,
169     OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED = 7501,
170     OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID = 7502,
171     OPTIX_ERROR_NOT_SUPPORTED = 7800,
172     OPTIX_ERROR_UNSUPPORTED_ABI_VERSION = 7801,
173     OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH = 7802,
174     OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS = 7803,
175     OPTIX_ERROR_LIBRARY_NOT_FOUND = 7804,
176     OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND = 7805,
177     OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE = 7806,
178     OPTIX_ERROR_DEVICE_OUT_OF_MEMORY = 7807,
179     OPTIX_ERROR_INVALID_POINTER = 7808,
180     OPTIX_ERROR_SYMBOL_NOT_FOUND = 7809,
181     OPTIX_ERROR_CUDA_ERROR = 7900,
182     OPTIX_ERROR_INTERNAL_ERROR = 7990,
183     OPTIX_ERROR_UNKNOWN = 7999,
184 } OptixResult;
185
189 typedef enum OptixDeviceProperty
190 {
192     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001,
193
196     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002,
197
200     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003,
201
204     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004,
205
208     OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005,
209
211     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006,
212
215     OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007,

```

```

216
219     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008,
220
224     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009,
225
229     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING = 0x200A,
230
234     OPTIX_DEVICE_PROPERTY_COOP_VEC = 0x200B,
235
239     OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL = 0x2020,
240
243     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_VERTICES = 0x2021,
244
247     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_TRIANGLES = 0x2022,
248
251     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_STRUCTURED_GRID_RESOLUTION = 0x2023,
252
255     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_SBT_INDEX = 0x2024,
256
259     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTERS_PER_GAS = 0x2025,
260 } OptixDeviceProperty;
261
286 typedef void (*OptixLogCallback)(unsigned int level, const char* tag, const char* message, void* cbdata);
287
295 typedef enum OptixDeviceContextValidationMode
296 {
297     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0,
298     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF
299 } OptixDeviceContextValidationMode;
300
304 typedef struct OptixDeviceContextOptions
305 {
307     OptixLogCallback logCallbackFunction;
309     void* logCallbackData;
311     int logCallbackLevel;
313     OptixDeviceContextValidationMode validationMode;
314 } OptixDeviceContextOptions;
315
320 typedef enum OptixPipelineSymbolMemcpyKind
321 {
322     OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_FROM_DEVICE = 0x21A0,
323     OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_FROM_HOST = 0x21A1,
324     OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_TO_DEVICE = 0x21A2,
325     OPTIX_PIPELINE_SYMBOL_MEMCPY_KIND_TO_HOST = 0x21A3,
326 } OptixPipelineSymbolMemcpyKind;
327
332 typedef enum OptixDevicePropertyShaderExecutionReorderingFlags
333 {
336     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_NONE = 0,
337
338     // Standard thread reordering is supported
339     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD = 1 < 0,
340 } OptixDevicePropertyShaderExecutionReorderingFlags;
341
346 typedef enum OptixDevicePropertyClusterAccelFlags
347 {
349     OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_NONE = 0,
350
351     // Cluster acceleration structure builds are supported.
352     OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_STANDARD = 1 < 0,
353 } OptixDevicePropertyClusterAccelFlags;
354
358 typedef enum OptixGeometryFlags
359 {
361     OPTIX_GEOMETRY_FLAG_NONE = 0,
362
365     OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u < 0,

```

```

366
370     OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u « 1,
371
375     OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u « 2,
376 } OptixGeometryFlags;
377
383 typedef enum OptixHitKind
384 {
386     OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE,
388     OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF
389 } OptixHitKind;
390
392 typedef enum OptixIndicesFormat
393 {
395     OPTIX_INDICES_FORMAT_NONE = 0,
397     OPTIX_INDICES_FORMAT_UNSIGNED_BYTE3 = 0x2101,
399     OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102,
401     OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103
402 } OptixIndicesFormat;
403
405 typedef enum OptixVertexFormat
406 {
407     OPTIX_VERTEX_FORMAT_NONE = 0,
408     OPTIX_VERTEX_FORMAT_FLOAT3 = 0x2121,
409     OPTIX_VERTEX_FORMAT_FLOAT2 = 0x2122,
410     OPTIX_VERTEX_FORMAT_HALF3 = 0x2123,
411     OPTIX_VERTEX_FORMAT_HALF2 = 0x2124,
412     OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125,
413     OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126
414 } OptixVertexFormat;
415
417 typedef enum OptixTransformFormat
418 {
419     OPTIX_TRANSFORM_FORMAT_NONE = 0,
420     OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1,
421 } OptixTransformFormat;
422
424 typedef enum OptixOpacityMicromapFormat
425 {
427     OPTIX_OPACITY_MICROMAP_FORMAT_NONE = 0,
429     OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE = 1,
431     OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE = 2,
432 } OptixOpacityMicromapFormat;
433
435 typedef enum OptixOpacityMicromapArrayIndexingMode
436 {
438     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0,
441     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1,
445     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2,
446 } OptixOpacityMicromapArrayIndexingMode;
447
452 typedef struct OptixOpacityMicromapUsageCount
453 {
456     unsigned int count;
458     unsigned int subdivisionLevel;
460     OptixOpacityMicromapFormat format;
461 } OptixOpacityMicromapUsageCount;
462
463 typedef struct OptixBuildInputOpacityMicromap
464 {
466     OptixOpacityMicromapArrayIndexingMode indexingMode;
467
472     CUdeviceptr opacityMicromapArray;
473
483     CUdeviceptr indexBuffer;
484
487     unsigned int indexSizeInBytes;

```

```

488
491     unsigned int indexStrideInBytes;
492
494     unsigned int indexOffset;
495
497     unsigned int numMicromapUsageCounts;
500     const OptixOpacityMicromapUsageCount* micromapUsageCounts;
501 } OptixBuildInputOpacityMicromap;
502
503 typedef struct OptixRelocateInputOpacityMicromap
504 {
508     CUdeviceptr opacityMicromapArray;
509 } OptixRelocateInputOpacityMicromap;
510
511
512
516 typedef struct OptixBuildInputTriangleArray
517 {
525     const CUdeviceptr* vertexBuffers;
526
528     unsigned int numVertices;
529
531     OptixVertexFormat vertexFormat;
532
535     unsigned int vertexStrideInBytes;
536
540     CUdeviceptr indexBuffer;
541
543     unsigned int numIndexTriplets;
544
546     OptixIndicesFormat indexFormat;
547
550     unsigned int indexStrideInBytes;
551
555     CUdeviceptr preTransform;
556
560     const unsigned int* flags;
561
563     unsigned int numSbtRecords;
564
568     CUdeviceptr sbtIndexOffsetBuffer;
569
571     unsigned int sbtIndexOffsetSizeInBytes;
572
575     unsigned int sbtIndexOffsetStrideInBytes;
576
579     unsigned int primitiveIndexOffset;
580
582     OptixTransformFormat transformFormat;
583
585     OptixBuildInputOpacityMicromap opacityMicromap;
586
587 } OptixBuildInputTriangleArray;
588
592 typedef struct OptixRelocateInputTriangleArray
593 {
596     unsigned int numSbtRecords;
597
599     OptixRelocateInputOpacityMicromap opacityMicromap;
600 } OptixRelocateInputTriangleArray;
601
604 typedef enum OptixPrimitiveType
605 {
607     OPTIX_PRIMITIVE_TYPE_CUSTOM = 0x2500,
609     OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE = 0x2501,
611     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE = 0x2502,
613     OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR = 0x2503,

```

```

615     OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM           = 0x2504,
617     OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE    = 0x2505,
619     OPTIX_PRIMITIVE_TYPE_SPHERE                    = 0x2506,
621     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER        = 0x2507,
623     OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE_ROCAPS = 0x2508,
625     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE_ROCAPS = 0x2509,
627     OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM_ROCAPS   = 0x250A,
629     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER_ROCAPS = 0x250B,
631     OPTIX_PRIMITIVE_TYPE_TRIANGLE                  = 0x2531,
632 } OptixPrimitiveType;
633
637 typedef enum OptixPrimitiveTypeFlags
638 {
640     OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM                = 1 « 0,
642     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE = 1 « 1,
644     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE    = 1 « 2,
646     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR          = 1 « 3,
648     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM       = 1 « 4,
650     OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE = 1 « 5,
652     OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE                 = 1 « 6,
654     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER     = 1 « 7,
656     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE_ROCAPS = 1 « 8,
658     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE_ROCAPS = 1 « 9,
660     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM_ROCAPS = 1 « 10,
662     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER_ROCAPS = 1 « 11,
664     OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE              = 1 « 31,
665 } OptixPrimitiveTypeFlags;
666
669 typedef enum OptixCurveEndcapFlags
670 {
672     OPTIX_CURVE_ENDCAP_DEFAULT           = 0,
674     OPTIX_CURVE_ENDCAP_ON                = 1 « 0,
675 } OptixCurveEndcapFlags;
676
694 typedef struct OptixBuildInputCurveArray
695 {
698     OptixPrimitiveType curveType;
700     unsigned int numPrimitives;
701
706     const CUdeviceptr* vertexBuffers;
708     unsigned int numVertices;
711     unsigned int vertexStrideInBytes;
712
715     const CUdeviceptr* widthBuffers;
718     unsigned int widthStrideInBytes;
719
721     const CUdeviceptr* normalBuffers;
723     unsigned int normalStrideInBytes;
724
730     CUdeviceptr indexBuffer;
733     unsigned int indexStrideInBytes;
734
737     unsigned int flag;
738
741     unsigned int primitiveIndexOffset;
742
744     unsigned int endcapFlags;
745 } OptixBuildInputCurveArray;
746
759 typedef struct OptixBuildInputSphereArray
760 {
765     const CUdeviceptr* vertexBuffers;
766
769     unsigned int vertexStrideInBytes;
771     unsigned int numVertices;
772
775     const CUdeviceptr* radiusBuffers;

```

```

778 unsigned int radiusStrideInBytes;
781 int singleRadius;
782
786 const unsigned int* flags;
787
789 unsigned int numSbtRecords;
793 CUdeviceptr sbtIndexOffsetBuffer;
795 unsigned int sbtIndexOffsetSizeInBytes;
798 unsigned int sbtIndexOffsetStrideInBytes;
799
802 unsigned int primitiveIndexOffset;
803 } OptixBuildInputSphereArray;
804
806 typedef struct OptixAabb
807 {
808     float minX;
809     float minY;
810     float minZ;
811     float maxX;
812     float maxY;
813     float maxZ;
814 } OptixAabb;
815
819 typedef struct OptixBuildInputCustomPrimitiveArray
820 {
825     const CUdeviceptr* aabbBuffers;
826
829     unsigned int numPrimitives;
830
834     unsigned int strideInBytes;
835
839     const unsigned int* flags;
840
842     unsigned int numSbtRecords;
843
847     CUdeviceptr sbtIndexOffsetBuffer;
848
850     unsigned int sbtIndexOffsetSizeInBytes;
851
854     unsigned int sbtIndexOffsetStrideInBytes;
855
858     unsigned int primitiveIndexOffset;
859 } OptixBuildInputCustomPrimitiveArray;
860
864 typedef struct OptixBuildInputInstanceArray
865 {
873     CUdeviceptr instances;
874
876     unsigned int numInstances;
877
881     unsigned int instanceStride;
882 } OptixBuildInputInstanceArray;
883
887 typedef struct OptixRelocateInputInstanceArray
888 {
891     unsigned int numInstances;
892
898     CUdeviceptr traversableHandles;
899
900 } OptixRelocateInputInstanceArray;
901
905 typedef enum OptixBuildInputType
906 {
908     OPTIX_BUILD_INPUT_TYPE_TRIANGLES = 0x2141,
910     OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES = 0x2142,
912     OPTIX_BUILD_INPUT_TYPE_INSTANCES = 0x2143,
914     OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS = 0x2144,

```

```

916     OPTIX_BUILD_INPUT_TYPE_CURVES = 0x2145,
918     OPTIX_BUILD_INPUT_TYPE_SPHERES = 0x2146
919 } OptixBuildInputType;
920
926 typedef struct OptixBuildInput
927 {
929     OptixBuildInputType type;
930
931     union
932     {
933         char pad[1024];
935         OptixBuildInputTriangleArray triangleArray;
937         OptixBuildInputCurveArray curveArray;
939         OptixBuildInputSphereArray sphereArray;
941         OptixBuildInputCustomPrimitiveArray customPrimitiveArray;
943         OptixBuildInputInstanceArray instanceArray;
944     };
945 } OptixBuildInput;
946
950 typedef struct OptixRelocateInput
951 {
953     OptixBuildInputType type;
954
955     union
956     {
958         OptixRelocateInputInstanceArray instanceArray;
959
961         OptixRelocateInputTriangleArray triangleArray;
962     };
964 };
965 } OptixRelocateInput;
966
970 typedef enum OptixInstanceFlags
971 {
973     OPTIX_INSTANCE_FLAG_NONE = 0,
974
978     OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u « 0,
979
982     OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u « 1,
983
987     OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT = 1u « 2,
988
993     OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u « 3,
994
995
997     OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u « 4,
1001     OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS = 1u « 5,
1002
1003 } OptixInstanceFlags;
1004
1008 typedef struct OptixInstance
1009 {
1011     float transform[12];
1012
1014     unsigned int instanceId;
1015
1019     unsigned int sbtOffset;
1020
1023     unsigned int visibilityMask;
1024
1026     unsigned int flags;
1027
1029     OptixTraversableHandle traversableHandle;
1030
1032     unsigned int pad[2];
1033 } OptixInstance;
1034

```

```

1038 typedef enum OptixBuildFlags
1039 {
1041     OPTIX_BUILD_FLAG_NONE = 0,
1042
1045     OPTIX_BUILD_FLAG_ALLOW_UPDATE = 1u « 0,
1046
1047     OPTIX_BUILD_FLAG_ALLOW_COMPACTION = 1u « 1,
1048
1050     OPTIX_BUILD_FLAG_PREFER_FAST_TRACE = 1u « 2,
1051
1053     OPTIX_BUILD_FLAG_PREFER_FAST_BUILD = 1u « 3,
1054
1069     OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS = 1u « 4,
1070
1073     OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS = 1u « 5,
1074
1078     OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE = 1u « 6,
1079
1083     OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1u « 7,
1084 } OptixBuildFlags;
1085
1086
1088 typedef enum OptixOpacityMicromapFlags
1089 {
1090     OPTIX_OPACITY_MICROMAP_FLAG_NONE = 0,
1091
1093     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 « 0,
1094
1096     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 « 1,
1097 } OptixOpacityMicromapFlags;
1098
1100 typedef struct OptixOpacityMicromapDesc
1101 {
1103     unsigned int    byteOffset;
1105     unsigned short subdivisionLevel;
1107     unsigned short format;
1108 } OptixOpacityMicromapDesc;
1109
1114 typedef struct OptixOpacityMicromapHistogramEntry
1115 {
1117     unsigned int    count;
1119     unsigned int    subdivisionLevel;
1121     OptixOpacityMicromapFormat format;
1122 } OptixOpacityMicromapHistogramEntry;
1123
1125 typedef struct OptixOpacityMicromapArrayBuildInput
1126 {
1128     unsigned int flags;
1129
1131     CUdeviceptr inputBuffer;
1132
1135     CUdeviceptr perMicromapDescBuffer;
1136
1140     unsigned int perMicromapDescStrideInBytes;
1141
1143     unsigned int numMicromapHistogramEntries;
1146     const OptixOpacityMicromapHistogramEntry* micromapHistogramEntries;
1147 } OptixOpacityMicromapArrayBuildInput;
1148
1150 typedef struct OptixMicromapBufferSizes
1151 {
1152     size_t outputSizeInBytes;
1153     size_t tempSizeInBytes;
1154 } OptixMicromapBufferSizes;
1155
1157 typedef struct OptixMicromapBuffers
1158 {

```

```

1160     CUdeviceptr output;
1162     size_t outputSizeInBytes;
1164     CUdeviceptr temp;
1166     size_t tempSizeInBytes;
1167 } OptixMicromapBuffers;
1168
1169
1181 typedef enum OptixBuildOperation
1182 {
1184     OPTIX_BUILD_OPERATION_BUILD = 0x2161,
1186     OPTIX_BUILD_OPERATION_UPDATE = 0x2162,
1187 } OptixBuildOperation;
1188
1192 typedef enum OptixMotionFlags
1193 {
1194     OPTIX_MOTION_FLAG_NONE = 0,
1195     OPTIX_MOTION_FLAG_START_VANISH = 1u « 0,
1196     OPTIX_MOTION_FLAG_END_VANISH = 1u « 1
1197 } OptixMotionFlags;
1198
1203 typedef struct OptixMotionOptions
1204 {
1207     unsigned short numKeys;
1208
1210     unsigned short flags;
1211
1213     float timeBegin;
1214
1216     float timeEnd;
1217 } OptixMotionOptions;
1218
1222 typedef struct OptixAccelBuildOptions
1223 {
1225     unsigned int buildFlags;
1226
1233     OptixBuildOperation operation;
1234
1236     OptixMotionOptions motionOptions;
1237 } OptixAccelBuildOptions;
1238
1244 typedef struct OptixAccelBufferSizes
1245 {
1248     size_t outputSizeInBytes;
1249
1252     size_t tempSizeInBytes;
1253
1258     size_t tempUpdateSizeInBytes;
1259 } OptixAccelBufferSizes;
1260
1264 typedef enum OptixAccelPropertyType
1265 {
1267     OPTIX_PROPERTY_TYPE_COMPACTED_SIZE = 0x2181,
1268
1270     OPTIX_PROPERTY_TYPE_AABBS = 0x2182,
1271 } OptixAccelPropertyType;
1272
1276 typedef struct OptixAccelEmitDesc
1277 {
1279     CUdeviceptr result;
1280
1282     OptixAccelPropertyType type;
1283 } OptixAccelEmitDesc;
1284
1289 typedef struct OptixRelocationInfo
1290 {
1292     unsigned long long info[4];
1293 } OptixRelocationInfo;

```

```

1294
1300 typedef struct OptixStaticTransform
1301 {
1303     OptixTraversableHandle child;
1304
1306     unsigned int pad[2];
1307
1309     float transform[12];
1310
1313     float invTransform[12];
1314 } OptixStaticTransform;
1315
1340 typedef struct OptixMatrixMotionTransform
1341 {
1343     OptixTraversableHandle child;
1344
1347     OptixMotionOptions motionOptions;
1348
1350     unsigned int pad[3];
1351
1353     float transform[2][12];
1354 } OptixMatrixMotionTransform;
1355
1363 //      [ sx  a  b  pvx ]
1364 //  S = [  0 sy  c  pvy ]
1365 //      [  0  0 sz  pvz ]
1374 //      [  1  0  0 tx ]
1375 //  T = [  0  1  0 ty ]
1376 //      [  0  0  1 tz ]
1386 typedef struct OptixSRTData
1387 {
1390     float sx, a, b, pvx, sy, c, pvy, sz, pvz, qx, qy, qz, qw, tx, ty, tz;
1392 } OptixSRTData;
1393
1394
1419 typedef struct OptixSRTMotionTransform
1420 {
1422     OptixTraversableHandle child;
1423
1426     OptixMotionOptions motionOptions;
1427
1429     unsigned int pad[3];
1430
1432     OptixSRTData srtData[2];
1433 } OptixSRTMotionTransform;
1434
1438 typedef enum OptixTraversableType
1439 {
1441     OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM = 0x21C1,
1443     OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM = 0x21C2,
1445     OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM = 0x21C3,
1446 } OptixTraversableType;
1447
1448
1454
1456 typedef enum OptixClusterAccelBuildFlags
1457 {
1458     OPTIX_CLUSTER_ACCEL_BUILD_FLAG_NONE = 0,
1459     OPTIX_CLUSTER_ACCEL_BUILD_FLAG_PREFER_FAST_TRACE = 1 « 0,
1460     OPTIX_CLUSTER_ACCEL_BUILD_FLAG_PREFER_FAST_BUILD = 1 « 1,
1461     OPTIX_CLUSTER_ACCEL_BUILD_FLAG_ALLOW_OPACITY_MICROMAPS = 1 « 2
1462 } OptixClusterAccelBuildFlags;
1463
1465 typedef enum OptixClusterAccelClusterFlags
1466 {
1467     OPTIX_CLUSTER_ACCEL_CLUSTER_FLAG_NONE = 0,
1470     OPTIX_CLUSTER_ACCEL_CLUSTER_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1 « 0,

```

```

1471 } OptixClusterAccelClusterFlags;
1472
1475 typedef enum OptixClusterAccelPrimitiveFlags
1476 {
1477     OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_NONE = 0,
1478     OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1 « 0,
1479     OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1 « 1,
1480     OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_ANYHIT = 1 « 2,
1481 } OptixClusterAccelPrimitiveFlags;
1482
1484 typedef enum OptixClusterAccelBuildType
1485 {
1486     OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS = 0x2545,
1487     OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES = 0x2546,
1488     OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES = 0x2547,
1489     OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES = 0x2548,
1490     OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS = 0x2549
1491 } OptixClusterAccelBuildType;
1492
1494 typedef enum OptixClusterAccelBuildMode
1495 {
1497     OPTIX_CLUSTER_ACCEL_BUILD_MODE_IMPLICIT_DESTINATIONS = 0,
1499     OPTIX_CLUSTER_ACCEL_BUILD_MODE_EXPLICIT_DESTINATIONS = 1,
1501     OPTIX_CLUSTER_ACCEL_BUILD_MODE_GET_SIZES = 2
1502 } OptixClusterAccelBuildMode;
1503
1505 typedef enum OptixClusterAccelIndicesFormat
1506 {
1507     OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_8BIT = 1,
1508     OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_16BIT = 2,
1509     OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_32BIT = 4,
1510 } OptixClusterAccelIndicesFormat;
1511
1512 typedef struct OptixClusterAccelBuildModeDescImplicitDest
1513 {
1518     CUdeviceptr    outputBuffer;
1520     size_t         outputBufferSizeInBytes;
1522     CUdeviceptr    tempBuffer;
1523     size_t         tempBufferSizeInBytes;
1524
1526     CUdeviceptr    outputHandlesBuffer;
1528     unsigned int   outputHandlesStrideInBytes;
1530     CUdeviceptr    outputSizesBuffer;
1532     unsigned int   outputSizesStrideInBytes;
1533 } OptixClusterAccelBuildModeDescImplicitDest;
1534
1535 typedef struct OptixClusterAccelBuildModeDescExplicitDest
1536 {
1538     CUdeviceptr    tempBuffer;
1539     size_t         tempBufferSizeInBytes;
1541     CUdeviceptr    destAddressesBuffer;
1543     unsigned int   destAddressesStrideInBytes;
1544
1546     CUdeviceptr    outputHandlesBuffer;
1548     unsigned int   outputHandlesStrideInBytes;
1550     CUdeviceptr    outputSizesBuffer;
1552     unsigned int   outputSizesStrideInBytes;
1553 } OptixClusterAccelBuildModeDescExplicitDest;
1554
1555 typedef struct OptixClusterAccelBuildModeDescGetSize
1556 {
1558     CUdeviceptr    outputSizesBuffer;
1560     unsigned int   outputSizesStrideInBytes;
1562     CUdeviceptr    tempBuffer;
1563     size_t         tempBufferSizeInBytes;
1564 } OptixClusterAccelBuildModeDescGetSize;
1565

```

```

1566 typedef struct OptixClusterAccelBuildInputTriangles
1567 {
1568     OptixClusterAccelBuildFlags flags;
1569
1573     unsigned int maxArgCount;
1575     OptixVertexFormat vertexFormat;
1579     unsigned int maxSbtIndexValue;
1581     unsigned int maxUniqueSbtIndexCountPerArg;
1582
1584     unsigned int maxTriangleCountPerArg;
1586     unsigned int maxVertexCountPerArg;
1588     unsigned int maxTotalTriangleCount;
1590     unsigned int maxTotalVertexCount;
1592     unsigned int minPositionTruncateBitCount;
1593 } OptixClusterAccelBuildInputTriangles;
1594
1595 typedef struct OptixClusterAccelBuildInputGrids
1596 {
1597     OptixClusterAccelBuildFlags flags;
1598     // Max number of OptixClusterAccelBuildInputGridsArgs provided at build time for
1599     OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRID
1600     unsigned int maxArgCount;
1601
1602     OptixVertexFormat vertexFormat;
1606     unsigned int maxSbtIndexValue;
1607
1608
1610     unsigned int maxWidth;
1612     unsigned int maxHeight;
1613 } OptixClusterAccelBuildInputGrids;
1614
1615 typedef struct OptixClusterAccelBuildInputClusters
1616 {
1617     OptixClusterAccelBuildFlags flags;
1619     unsigned int maxArgCount;
1620
1621     unsigned int maxTotalClusterCount;
1622     unsigned int maxClusterCountPerArg;
1623 } OptixClusterAccelBuildInputClusters;
1624
1625 typedef struct OptixClusterAccelPrimitiveInfo
1626 {
1627     unsigned int sbtIndex : 24;
1628     unsigned int reserved : 5;
1630     unsigned int primitiveFlags : 3;
1631 } OptixClusterAccelPrimitiveInfo;
1632
1634 typedef enum OptixClusterIDValues {
1635     OPTIX_CLUSTER_ID_INVALID = 0xFFFFFFFFu,
1636 } OptixClusterIDValues;
1637
1639 typedef struct OptixClusterAccelBuildInputTrianglesArgs
1640 {
1643     unsigned int clusterId;
1645     unsigned int clusterFlags;
1646
1647     // Packing the following values into a single 32b value
1649     unsigned int triangleCount : 9;
1651     unsigned int vertexCount : 9;
1654     unsigned int positionTruncateBitCount : 6;
1656     unsigned int indexFormat : 4;
1658     unsigned int opacityMicromapIndexFormat : 4;
1659
1661     OptixClusterAccelPrimitiveInfo basePrimitiveInfo;
1662
1664     unsigned short indexBufferStrideInBytes;
1666     unsigned short vertexBufferStrideInBytes;

```

```

1668     unsigned short primitiveInfoBufferStrideInBytes;
1670     unsigned short opacityMicromapIndexBufferStrideInBytes;
1671
1673     CUdeviceptr indexBuffer;
1679     CUdeviceptr vertexBuffer;
1681     CUdeviceptr primitiveInfoBuffer;
1683     CUdeviceptr opacityMicromapArray;
1685     CUdeviceptr opacityMicromapIndexBuffer;
1686
1691     CUdeviceptr instantiationBoundingBoxLimit;
1692 } OptixClusterAccelBuildInputTrianglesArgs;
1693
1695 typedef struct OptixClusterAccelBuildInputGridsArgs
1696 {
1699     unsigned int baseClusterId;
1701     unsigned int clusterFlags;
1702
1704     OptixClusterAccelPrimitiveInfo basePrimitiveInfo;
1705
1706     // Packing the following values into a single 32b value
1708     unsigned int positionTruncateBitCount : 6;
1709     unsigned int reserved                  : 26;
1710
1711     // Packing the following values into a single 32b value
1713     unsigned char dimensions[2];
1714     unsigned short reserved2;
1715 } OptixClusterAccelBuildInputGridsArgs;
1716
1718 typedef struct OptixClusterAccelBuildInputTemplatesArgs
1719 {
1721     unsigned int clusterIdOffset;
1722
1725     unsigned int sbtIndexOffset;
1726
1728     CUdeviceptr clusterTemplate;
1731     CUdeviceptr vertexBuffer;
1733     unsigned int vertexStrideInBytes;
1734     unsigned int reserved;
1735 } OptixClusterAccelBuildInputTemplatesArgs;
1736
1738 typedef struct OptixClusterAccelBuildInputClustersArgs
1739 {
1741     unsigned int clusterHandlesCount;
1742     unsigned int clusterHandlesBufferStrideInBytes;
1746     CUdeviceptr clusterHandlesBuffer;
1747 } OptixClusterAccelBuildInputClustersArgs;
1748
1749 typedef struct OptixClusterAccelBuildInput
1750 {
1751     OptixClusterAccelBuildType type;
1752
1753     union
1754     {
1757         OptixClusterAccelBuildInputTriangles triangles;
1759         OptixClusterAccelBuildInputClusters clusters;
1761         OptixClusterAccelBuildInputGrids grids;
1762     };
1763 } OptixClusterAccelBuildInput;
1764
1765 typedef struct OptixClusterAccelBuildModeDesc
1766 {
1767     OptixClusterAccelBuildMode mode;
1768     union
1769     {
1770         OptixClusterAccelBuildModeDescImplicitDest implicitDest;
1771         OptixClusterAccelBuildModeDescExplicitDest explicitDest;
1772         OptixClusterAccelBuildModeDescGetSize      getSize;

```

```

1773     };
1774 } OptixClusterAccelBuildModeDesc;
1775
1776
1782
1783
1787 typedef enum OptixPixelFormat
1788 {
1789     OPTIX_PIXEL_FORMAT_HALF1 = 0x220a,
1790     OPTIX_PIXEL_FORMAT_HALF2 = 0x2207,
1791     OPTIX_PIXEL_FORMAT_HALF3 = 0x2201,
1792     OPTIX_PIXEL_FORMAT_HALF4 = 0x2202,
1793     OPTIX_PIXEL_FORMAT_FLOAT1 = 0x220b,
1794     OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208,
1795     OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203,
1796     OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204,
1797     OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205,
1798     OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206,
1799     OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209
1800 } OptixPixelFormat;
1801
1805 typedef struct OptixImage2D
1806 {
1808     CUdeviceptr data;
1810     unsigned int width;
1812     unsigned int height;
1814     unsigned int rowStrideInBytes;
1819     unsigned int pixelStrideInBytes;
1821     OptixPixelFormat format;
1822 } OptixImage2D;
1823
1827 typedef enum OptixDenoiserModelKind
1828 {
1830     OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324,
1831
1833     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326,
1834
1836     OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327,
1837
1839     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328,
1840
1843     OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322,
1844     OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323,
1845
1847     OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325
1848
1849 } OptixDenoiserModelKind;
1850
1854 typedef enum OptixDenoiserAlphaMode
1855 {
1857     OPTIX_DENOISER_ALPHA_MODE_COPY = 0,
1858
1860     OPTIX_DENOISER_ALPHA_MODE_DENOISE = 1
1861 } OptixDenoiserAlphaMode;
1862
1866 typedef struct OptixDenoiserOptions
1867 {
1868     // if nonzero, albedo image must be given in OptixDenoiserGuideLayer
1869     unsigned int guideAlbedo;
1870
1871     // if nonzero, normal image must be given in OptixDenoiserGuideLayer
1872     unsigned int guideNormal;
1873
1875     OptixDenoiserAlphaMode denoiseAlpha;
1876 } OptixDenoiserOptions;
1877
1881 typedef struct OptixDenoiserGuideLayer

```

```

1882 {
1883     // image with three components: R, G, B.
1884     OptixImage2D albedo;
1885
1886     // image with two or three components: X, Y, Z.
1887     // (X, Y) camera space for OPTIX_DENOISER_MODEL_KIND_LDR, OPTIX_DENOISER_MODEL_KIND_HDR models.
1888     // (X, Y, Z) world space, all other models.
1889     OptixImage2D normal;
1890
1891     // image with two components: X, Y.
1892     // pixel movement from previous to current frame for each pixel in screen space.
1893     OptixImage2D flow;
1894
1895     // Internal images used in temporal AOV denoising modes,
1896     // pixel format OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER.
1897     OptixImage2D previousOutputInternalGuideLayer;
1898     OptixImage2D outputInternalGuideLayer;
1899
1900     // image with a single component value that specifies how trustworthy the flow vector at x,y
position in
1901     // OptixDenoiserGuideLayer::flow is. Range 0..1 (low->high trustworthiness).
1902     // Ignored if data pointer in the image is zero.
1903     OptixImage2D flowTrustworthiness;
1904
1905 } OptixDenoiserGuideLayer;
1906
1907 typedef enum OptixDenoiserAOVType
1908 {
1909     OPTIX_DENOISER_AOV_TYPE_NONE          = 0,
1910
1911     OPTIX_DENOISER_AOV_TYPE_BEAUTY        = 0x7000,
1912     OPTIX_DENOISER_AOV_TYPE_SPECULAR      = 0x7001,
1913     OPTIX_DENOISER_AOV_TYPE_REFLECTION    = 0x7002,
1914     OPTIX_DENOISER_AOV_TYPE_REFRACTION    = 0x7003,
1915     OPTIX_DENOISER_AOV_TYPE_DIFFUSE       = 0x7004
1916 } OptixDenoiserAOVType;
1917
1918 typedef struct OptixDenoiserLayer
1919 {
1920     // input image (beauty or AOV)
1921     OptixImage2D input;
1922
1923     // denoised output image from previous frame if temporal model kind selected
1924     OptixImage2D previousOutput;
1925
1926     // denoised output for given input
1927     OptixImage2D output;
1928
1929     // Type of AOV, used in temporal AOV modes as a hint to improve image quality.
1930     OptixDenoiserAOVType type;
1931 } OptixDenoiserLayer;
1932
1933 typedef struct OptixDenoiserParams
1934 {
1935     CUdeviceptr  hdrIntensity;
1936
1937     float         blendFactor;
1938
1939     CUdeviceptr  hdrAverageColor;
1940
1941     unsigned int  temporalModeUsePreviousLayers;
1942
1943     float         flowMulX;
1944     float         flowMulY;
1945 } OptixDenoiserParams;

```

```

1978
1982 typedef struct OptixDenoiserSizes
1983 {
1985     size_t stateSizeInBytes;
1986
1989     size_t withOverlapScratchSizeInBytes;
1990
1993     size_t withoutOverlapScratchSizeInBytes;
1994
1996     unsigned int overlapWindowSizeInPixels;
1997
2000     size_t computeAverageColorSizeInBytes;
2001
2004     size_t computeIntensitySizeInBytes;
2005
2007     size_t internalGuideLayerPixelSizeInBytes;
2008 } OptixDenoiserSizes;
2009
2010
2016
2017
2022 typedef enum OptixRayFlags
2023 {
2025     OPTIX_RAY_FLAG_NONE = 0u,
2026
2031     OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u « 0,
2032
2037     OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u « 1,
2038
2041     OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u « 2,
2042
2044     OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u « 3,
2045
2050     OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u « 4,
2051
2056     OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u « 5,
2057
2063     OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u « 6,
2064
2070     OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u « 7,
2071
2073     OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u « 10,
2074 } OptixRayFlags;
2075
2081 typedef enum OptixTransformType
2082 {
2083     OPTIX_TRANSFORM_TYPE_NONE = 0,
2084     OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM = 1,
2085     OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM = 2,
2086     OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM = 3,
2087     OPTIX_TRANSFORM_TYPE_INSTANCE = 4,
2088 } OptixTransformType;
2089
2095 typedef struct OptixTraverseData
2096 {
2097     unsigned int data[20];
2098 } OptixTraverseData;
2099
2102 typedef enum OptixTraversableGraphFlags
2103 {
2106     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0,
2107
2111     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u « 0,
2112
2117     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u « 1,
2118 } OptixTraversableGraphFlags;
2119

```

```

2123 typedef enum OptixCompileOptimizationLevel
2124 {
2126     OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0,
2128     OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340,
2130     OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341,
2132     OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342,
2134     OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343,
2135 } OptixCompileOptimizationLevel;
2136
2140 typedef enum OptixCompileDebugLevel
2141 {
2143     OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT = 0,
2145     OPTIX_COMPILE_DEBUG_LEVEL_NONE = 0x2350,
2148     OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL = 0x2351,
2150     OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353,
2152     OPTIX_COMPILE_DEBUG_LEVEL_FULL = 0x2352,
2153 } OptixCompileDebugLevel;
2154
2158 typedef enum OptixModuleCompileState
2159 {
2161     OPTIX_MODULE_COMPILE_STATE_NOT_STARTED = 0x2360,
2162
2164     OPTIX_MODULE_COMPILE_STATE_STARTED = 0x2361,
2165
2167     OPTIX_MODULE_COMPILE_STATE_PENDING_FAILURE = 0x2362,
2168
2170     OPTIX_MODULE_COMPILE_STATE_FAILED = 0x2363,
2171
2173     OPTIX_MODULE_COMPILE_STATE_COMPLETED = 0x2364,
2174 } OptixModuleCompileState;
2175
2186 typedef enum OptixCreationFlags
2187 {
2188     OPTIX_CREATION_FLAG_NONE = 0,
2189     OPTIX_CREATION_FLAG_BLOCK_UNTIL_EFFECTIVE = 1 « 0,
2190 } OptixCreationFlags;
2191
2192
2225 typedef struct OptixModuleCompileBoundValueEntry {
2226     size_t pipelineParamOffsetInBytes;
2227     size_t sizeInBytes;
2228     const void* boundValuePtr;
2229     const char* annotation; // optional string to display, set to 0 if unused. If unused,
2230                           // OptiX will report the annotation as "No annotation"
2231 } OptixModuleCompileBoundValueEntry;
2232
2234 typedef enum OptixPayloadTypeID {
2235     OPTIX_PAYLOAD_TYPE_DEFAULT = 0,
2236     OPTIX_PAYLOAD_TYPE_ID_0 = (1 « 0u),
2237     OPTIX_PAYLOAD_TYPE_ID_1 = (1 « 1u),
2238     OPTIX_PAYLOAD_TYPE_ID_2 = (1 « 2u),
2239     OPTIX_PAYLOAD_TYPE_ID_3 = (1 « 3u),
2240     OPTIX_PAYLOAD_TYPE_ID_4 = (1 « 4u),
2241     OPTIX_PAYLOAD_TYPE_ID_5 = (1 « 5u),
2242     OPTIX_PAYLOAD_TYPE_ID_6 = (1 « 6u),
2243     OPTIX_PAYLOAD_TYPE_ID_7 = (1 « 7u)
2244 } OptixPayloadTypeID;
2245
2259 typedef enum OptixPayloadSemantics
2260 {
2261     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE = 0,
2262     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ = 1u « 0,
2263     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE = 2u « 0,
2264     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u « 0,
2265
2266     OPTIX_PAYLOAD_SEMANTICS_CH_NONE = 0,
2267     OPTIX_PAYLOAD_SEMANTICS_CH_READ = 1u « 2,

```

```

2268     OPTIX_PAYLOAD_SEMANTICS_CH_WRITE           = 2u « 2,
2269     OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE      = 3u « 2,
2270
2271     OPTIX_PAYLOAD_SEMANTICS_MS_NONE            = 0,
2272     OPTIX_PAYLOAD_SEMANTICS_MS_READ            = 1u « 4,
2273     OPTIX_PAYLOAD_SEMANTICS_MS_WRITE          = 2u « 4,
2274     OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE      = 3u « 4,
2275
2276     OPTIX_PAYLOAD_SEMANTICS_AH_NONE            = 0,
2277     OPTIX_PAYLOAD_SEMANTICS_AH_READ            = 1u « 6,
2278     OPTIX_PAYLOAD_SEMANTICS_AH_WRITE          = 2u « 6,
2279     OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE      = 3u « 6,
2280
2281     OPTIX_PAYLOAD_SEMANTICS_IS_NONE            = 0,
2282     OPTIX_PAYLOAD_SEMANTICS_IS_READ            = 1u « 8,
2283     OPTIX_PAYLOAD_SEMANTICS_IS_WRITE          = 2u « 8,
2284     OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE      = 3u « 8,
2285 } OptixPayloadSemantics;
2286
2287 typedef struct OptixPayloadType
2288 {
2291     unsigned int numPayloadValues;
2292
2294     const unsigned int *payloadSemantics;
2295 } OptixPayloadType;
2296
2300 typedef struct OptixModuleCompileOptions
2301 {
2304     int maxRegisterCount;
2305
2307     OptixCompileOptimizationLevel optLevel;
2308
2310     OptixCompileDebugLevel debugLevel;
2311
2313     const OptixModuleCompileBoundValueEntry* boundValues;
2314
2316     unsigned int numBoundValues;
2317
2320     unsigned int numPayloadTypes;
2321
2323     const OptixPayloadType* payloadTypes;
2324
2326     OptixModule baseModule;
2327 } OptixModuleCompileOptions;
2328
2333 typedef struct OptixBuiltinISOptions
2334 {
2335     OptixPrimitiveType      builtinISModuleType;
2337     int                     usesMotionBlur;
2339     unsigned int            buildFlags;
2341     unsigned int            curveEndcapFlags;
2342 } OptixBuiltinISOptions;
2343
2345 typedef enum OptixProgramGroupKind
2346 {
2349     OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421,
2350
2353     OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422,
2354
2357     OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423,
2358
2361     OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424,
2362
2365     OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425
2366 } OptixProgramGroupKind;
2367
2369 typedef enum OptixProgramGroupFlags

```

```

2370 {
2372     OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0
2373 } OptixProgramGroupFlags;
2374
2381 typedef struct OptixProgramGroupSingleModule
2382 {
2384     OptixModule module;
2386     const char* entryFunctionName;
2387 } OptixProgramGroupSingleModule;
2388
2394 typedef struct OptixProgramGroupHitgroup
2395 {
2397     OptixModule moduleCH;
2399     const char* entryFunctionNameCH;
2401     OptixModule moduleAH;
2403     const char* entryFunctionNameAH;
2405     OptixModule moduleIS;
2407     const char* entryFunctionNameIS;
2408 } OptixProgramGroupHitgroup;
2409
2415 typedef struct OptixProgramGroupCallables
2416 {
2418     OptixModule moduleDC;
2420     const char* entryFunctionNameDC;
2422     OptixModule moduleCC;
2424     const char* entryFunctionNameCC;
2425 } OptixProgramGroupCallables;
2426
2428 typedef struct OptixProgramGroupDesc
2429 {
2431     OptixProgramGroupKind kind;
2432
2434     unsigned int flags;
2435
2436     union
2437     {
2439         OptixProgramGroupHitgroup hitgroup;
2441         OptixProgramGroupSingleModule raygen;
2443         OptixProgramGroupSingleModule miss;
2445         OptixProgramGroupSingleModule exception;
2447         OptixProgramGroupCallables callables;
2448     };
2449 } OptixProgramGroupDesc;
2450
2454 typedef struct OptixProgramGroupOptions
2455 {
2468     const OptixPayloadType* payloadType;
2469 } OptixProgramGroupOptions;
2470
2472 typedef enum OptixExceptionCodes
2473 {
2476     OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1,
2477
2480     OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2,
2481
2482
2483 } OptixExceptionCodes;
2484
2488 typedef enum OptixExceptionFlags
2489 {
2491     OPTIX_EXCEPTION_FLAG_NONE = 0,
2492
2499     OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u << 0,
2500
2507     OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u << 1,
2508
2511     OPTIX_EXCEPTION_FLAG_USER = 1u << 2,

```

```

2512
2513 } OptixExceptionFlags;
2514
2520 typedef struct OptixPipelineCompileOptions
2521 {
2522     int usesMotionBlur;
2524
2526     unsigned int traversableGraphFlags;
2527
2530     int numPayloadValues;
2531
2534     int numAttributeValues;
2535
2537     unsigned int exceptionFlags;
2538
2542     const char* pipelineLaunchParamsVariableName;
2543
2547     size_t pipelineLaunchParamsSizeInBytes;
2548
2551     unsigned int usesPrimitiveTypeFlags;
2552
2554     int allowOpacityMicromaps;
2555
2558     int allowClusteredGeometry;
2559 } OptixPipelineCompileOptions;
2560
2564 typedef struct OptixPipelineLinkOptions
2565 {
2568     unsigned int maxTraceDepth;
2569
2572     unsigned int maxContinuationCallableDepth;
2576     unsigned int maxDirectCallableDepthFromState;
2580     unsigned int maxDirectCallableDepthFromTraversal;
2581
2586     unsigned int maxTraversableGraphDepth;
2587 } OptixPipelineLinkOptions;
2588
2592 typedef struct OptixShaderBindingTable
2593 {
2596     CUdeviceptr raygenRecord;
2597
2600     CUdeviceptr exceptionRecord;
2601
2605     CUdeviceptr missRecordBase;
2606     unsigned int missRecordStrideInBytes;
2607     unsigned int missRecordCount;
2609
2613     CUdeviceptr hitgroupRecordBase;
2614     unsigned int hitgroupRecordStrideInBytes;
2615     unsigned int hitgroupRecordCount;
2617
2622     CUdeviceptr callablesRecordBase;
2623     unsigned int callablesRecordStrideInBytes;
2624     unsigned int callablesRecordCount;
2626
2627 } OptixShaderBindingTable;
2628
2632 typedef struct OptixStackSizes
2633 {
2635     unsigned int cssRG;
2637     unsigned int cssMS;
2639     unsigned int cssCH;
2641     unsigned int cssAH;
2643     unsigned int cssIS;
2645     unsigned int cssCC;
2647     unsigned int dssDC;
2648

```

```

2649 } OptixStackSizes;
2650
2651
2652
2653
2662 typedef enum OptixDevicePropertyCoopVecFlags
2663 {
2664     OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_NONE      = 0,
2665
2666     // Standard cooperative vector features are supported
2667     OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_STANDARD = 1 « 0,
2670 } OptixDevicePropertyCoopVecFlags;
2671
2672 typedef enum OptixCoopVecElemType
2673 {
2674     OPTIX_COOP_VEC_ELEM_TYPE_UNKNOWN = 0x2A00,
2675     OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16 = 0x2A01,
2676     OPTIX_COOP_VEC_ELEM_TYPE_FLOAT32 = 0x2A03,
2677     OPTIX_COOP_VEC_ELEM_TYPE_UINT8   = 0x2A04,
2678     OPTIX_COOP_VEC_ELEM_TYPE_INT8    = 0x2A05,
2679     OPTIX_COOP_VEC_ELEM_TYPE_UINT32  = 0x2A08,
2680     OPTIX_COOP_VEC_ELEM_TYPE_INT32   = 0x2A09,
2681     OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E4M3 = 0x2A0A,
2682     OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E5M2 = 0x2A0B,
2690 } OptixCoopVecElemType;
2691
2692
2693 typedef enum OptixCoopVecMatrixLayout
2694 {
2695     OPTIX_COOP_VEC_MATRIX_LAYOUT_ROW_MAJOR      = 0x2A40,
2696     OPTIX_COOP_VEC_MATRIX_LAYOUT_COLUMN_MAJOR = 0x2A41,
2697     OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL = 0x2A42,
2698     OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL   = 0x2A43,
2699 } OptixCoopVecMatrixLayout;
2700
2701 typedef struct OptixCoopVecMatrixDescription
2702 {
2703     unsigned int      N;
2704     unsigned int      K;
2705     unsigned int      offsetInBytes;
2706     OptixCoopVecElemType elementType;
2707     OptixCoopVecMatrixLayout layout;
2708     unsigned int      rowColumnStrideInBytes;
2709     unsigned int      sizeInBytes;
2716 } OptixCoopVecMatrixDescription;
2717
2718 typedef struct OptixNetworkDescription
2719 {
2720     OptixCoopVecMatrixDescription* layers;
2721     unsigned int                  numLayers;
2722 } OptixNetworkDescription;
2723
2724
2725
2726
2732 typedef enum OptixQueryFunctionTableOptions
2733 {
2734     OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0
2735 } OptixQueryFunctionTableOptions;
2736
2737
2738
2739
2740 typedef OptixResult(OptixQueryFunctionTable_t)(int      abiId,
2741                                                unsigned int numOptions,
2742                                                OptixQueryFunctionTableOptions* /*optionKeys*/,
2743                                                const void** /*optionValues*/,
2744                                                void*      functionTable,
2745                                                size_t      sizeOfTable);
2746
2747
2748 // end group optix_types

```

```
2750
2751 #endif // OPTIX_OPTIX_TYPES_H
```

8.27 main.dox File Reference