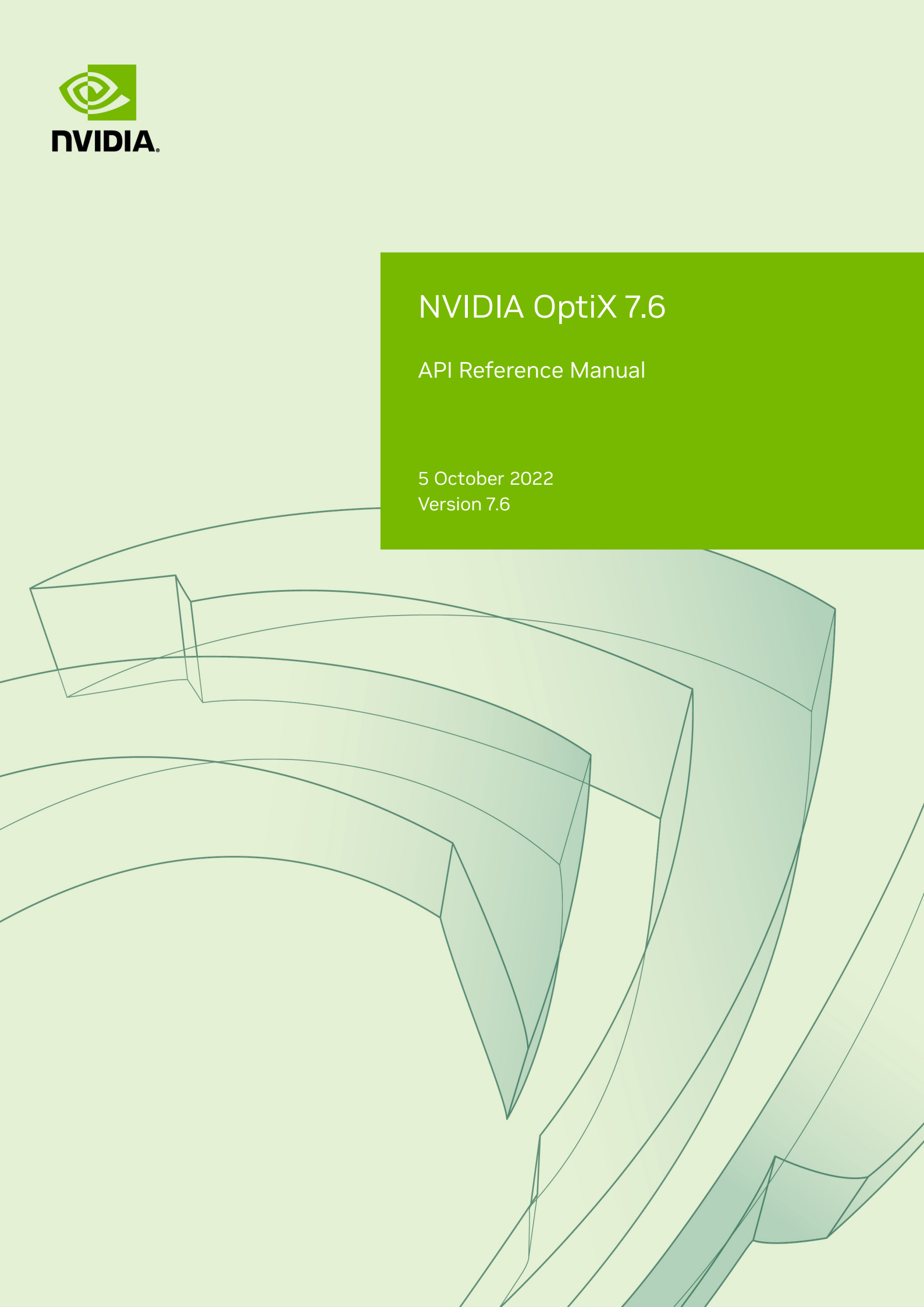

NVIDIA OptiX 7.6

API Reference Manual

5 October 2022
Version 7.6

Table of Contents

1 NVIDIA OptiX [rtmain] API

This document describes the OptiX 7.6 application programming interface. For more information about programming with OptiX 7.6, see `https://raytracing-docs.nvidia.com/`.

2 Module Index

2.1 Modules

Here is a list of all modules:

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

5 Module Documentation

5.1 Device API

Functions

- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_5 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_6 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_7 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_8 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_9 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_10 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_11 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_12 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_13 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_14 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_15 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_16 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_17 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_18 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_19 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_20 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_21 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_22 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_23 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_24 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_25 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_26 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_27 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_28 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_29 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_30 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_31 (unsigned int p)

- static __forceinline__ __device__ unsigned int optixGetPayload_0 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_1 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_2 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_3 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_4 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_5 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_6 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_7 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_8 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_9 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_10 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_11 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_12 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_13 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_14 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_15 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_16 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_17 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_18 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_19 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_20 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_21 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_22 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_23 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_24 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_25 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_26 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_27 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_28 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_29 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_30 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_31 ()
- static __forceinline__ __device__ void optixSetPayloadTypes (unsigned int typeMask)
- static __forceinline__ __device__ unsigned int optixUndefinedValue ()
- static __forceinline__ __device__ float3 optixGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixGetObjectRayOrigin ()
- static __forceinline__ __device__ float3 optixGetObjectRayDirection ()
- static __forceinline__ __device__ float optixGetRayTmin ()
- static __forceinline__ __device__ float optixGetRayTmax ()
- static __forceinline__ __device__ float optixGetRayTime ()
- static __forceinline__ __device__ unsigned int optixGetRayFlags ()
- static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceTraversableFromIAS (OptixTraversableHandle ias, unsigned int instIdx)
- static __forceinline__ __device__ void optixGetTriangleVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static __forceinline__ __device__ void optixGetLinearCurveVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])

- static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetCubicBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetSphereData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle ()
- static __forceinline__ __device__ float optixGetGASMotionTimeBegin (OptixTraversableHandle gas)
- static __forceinline__ __device__ float optixGetGASMotionTimeEnd (OptixTraversableHandle gas)
- static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (OptixTraversableHandle gas)
- static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (float m[12])
- static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (float3 normal)
- static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (float3 normal)
- static __forceinline__ __device__ unsigned int optixGetTransformListSize ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTransformType optixGetTransformTypeFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixStaticTransform * optixGetStaticTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixSRTMotionTransform * optixGetSRTMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixMatrixMotionTransform * optixGetMatrixMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceChildFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceInverseTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0)

- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6, unsigned int a7)
- static __forceinline__ __device__ unsigned int optixGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_7 ()
- static __forceinline__ __device__ void optixTerminateRay ()
- static __forceinline__ __device__ void optixIgnoreIntersection ()
- static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixGetHitKind ()
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsFrontFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsBackFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ()
- static __forceinline__ __device__ bool optixIsFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsBackFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleHit ()
- static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ()
- static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ()
- static __forceinline__ __device__ float optixGetCurveParameter ()
- static __forceinline__ __device__ uint3 optixGetLaunchIndex ()
- static __forceinline__ __device__ uint3 optixGetLaunchDimensions ()
- static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ()
- static __forceinline__ __device__ void optixThrowException (int exceptionCode)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0)

- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6, unsigned int exceptionDetail7)
- static __forceinline__ __device__ int optixGetExceptionCode ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7 ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable ()
- static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset ()
- static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay ()
- static __forceinline__ __device__ OptixParameterMismatchExceptionDetails optixGetExceptionParameterMismatch ()
- static __forceinline__ __device__ char * optixGetExceptionLineInfo ()
- template<typename ReturnT , typename... ArgTypes> static __forceinline__ __device__ ReturnT optixDirectCall (unsigned int sbtIndex, ArgTypes... args)
- template<typename ReturnT , typename... ArgTypes> static __forceinline__ __device__ ReturnT optixContinuationCall (unsigned int sbtIndex, ArgTypes... args)
- static __forceinline__ __device__ uint4 optixTexFootprint2D (unsigned long long tex, unsigned int texInfo, float x, float y, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DLod (unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool coarse, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DGrad (unsigned long long tex, unsigned int texInfo, float x, float y, float dPdx_x, float dPdx_y, float dPdy_x, float dPdy_y, bool coarse, unsigned int *singleMipLevel)

5.1.1 Detailed Description

OptiX Device API.

5.1.2 Function Documentation

5.1.2.1 optixContinuationCall()

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

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

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

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

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

Parameters

in	*sbtIndex*	The offset of the SBT entry of the continuation callable program to call relative to OptixShaderBindingTable::callablesRecordBase.
in	*args*	The arguments to pass to the continuation callable program.

5.1.2.2 optixDirectCall()

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

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

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

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

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

Parameters

in	*sbtIndex*	The offset of the SBT entry of the direct callable program to call relative to OptixShaderBindingTable::callablesRecordBase.
in	*args*	The arguments to pass to the direct callable program.

5.1.2.3 optixGetAttribute_0()

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

Returns the attribute at slot 0.

5.1.2.4 optixGetAttribute_1()

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

Returns the attribute at slot 1.

5.1.2.5 optixGetAttribute_2()

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

Returns the attribute at slot 2.

5.1.2.6 optixGetAttribute_3()

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

Returns the attribute at slot 3.

5.1.2.7 optixGetAttribute_4()

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

Returns the attribute at slot 4.

5.1.2.8 optixGetAttribute_5()

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

Returns the attribute at slot 5.

5.1.2.9 optixGetAttribute_6()

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

Returns the attribute at slot 6.

5.1.2.10 optixGetAttribute_7()

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

Returns the attribute at slot 7.

5.1.2.11 optixGetCatmullRomVertexData()

```
static __forceinline__ __device__ void optixGetCatmullRomVertexData (
          OptixTraversableHandle gas,
```

```
unsigned int primIdx,
unsigned int sbtGASIndex,
float time,
float4 data[4] )  [static]
```

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

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

5.1.2.12 optixGetCubicBSplineVertexData()

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

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

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

5.1.2.13 optixGetCurveParameter()

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

Convenience function that returns the curve parameter.

When using OptixBuildInputCurveArray objects, during intersection the curve parameter is stored into the first attribute register.

5.1.2.14 optixGetExceptionCode()

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

Returns the exception code.

Only available in EX.

5.1.2.15 optixGetExceptionDetail_0()

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

Returns the 32-bit exception detail at slot 0.

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

Only available in EX.

5.1.2.16 optixGetExceptionDetail_1()

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

Returns the 32-bit exception detail at slot 1.

See also optixGetExceptionDetail_0( )

5.1.2.17 optixGetExceptionDetail_2()

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

Returns the 32-bit exception detail at slot 2.

See also optixGetExceptionDetail_0( )

5.1.2.18 optixGetExceptionDetail_3()

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

Returns the 32-bit exception detail at slot 3.

See also optixGetExceptionDetail_0( )

5.1.2.19 optixGetExceptionDetail_4()

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

Returns the 32-bit exception detail at slot 4.

See also optixGetExceptionDetail_0( )

5.1.2.20 optixGetExceptionDetail_5()

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

Returns the 32-bit exception detail at slot 5.

See also optixGetExceptionDetail_0( )

5.1.2.21 optixGetExceptionDetail_6()

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

Returns the 32-bit exception detail at slot 6.

See also optixGetExceptionDetail_0( )

5.1.2.22 optixGetExceptionDetail_7()

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

Returns the 32-bit exception detail at slot 7.

See also optixGetExceptionDetail_0( )

5.1.2.23 optixGetExceptionInvalidRay()

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

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

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

Only available in EX.

5.1.2.24 optixGetExceptionInvalidSbtOffset()

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

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

Returns zero for all other exception codes.

Only available in EX.

5.1.2.25 optixGetExceptionInvalidTraversable()

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

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

Returns zero for all other exception codes.

Only available in EX.

5.1.2.26 optixGetExceptionLineInfo()

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

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

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

Returns a NULL pointer if no line information is available.

Only available in EX.

5.1.2.27 optixGetExceptionParameterMismatch()

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

Returns information about an exception with code OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH.

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

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

Only available in EX.

5.1.2.28 optixGetGASMotionStepCount()

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

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

5.1.2.29 optixGetGASMotionTimeBegin()

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

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

5.1.2.30 optixGetGASMotionTimeEnd()

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

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

5.1.2.31 optixGetGASTraversableHandle()

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

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

5.1.2.32 optixGetHitKind()

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

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

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

Available only in AH and CH.

5.1.2.33 optixGetInstanceChildFromHandle()

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

Returns child traversable handle from an OptixInstance traversable.

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

5.1.2.34 optixGetInstanceId()

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

Returns the OptixInstance::instanceId of the instance within the top level acceleration structure associated with the current intersection.

When building an acceleration structure using OptixBuildInputInstanceArray each OptixInstance has a user supplied instanceId. OptixInstance objects reference another acceleration structure. During traversal the acceleration structures are visited top down. In the IS and AH programs the OptixInstance::instanceId corresponding to the most recently visited OptixInstance is returned when calling optixGetInstanceId( ). In CH optixGetInstanceId( ) returns the OptixInstance::instanceId when the hit was recorded with optixReportIntersection. In the case where there is no OptixInstance visited, optixGetInstanceId returns ~0u

5.1.2.35 optixGetInstanceIdFromHandle()

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

Returns instanceId from an OptixInstance traversable.

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

5.1.2.36 optixGetInstanceIndex()

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

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

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

5.1.2.37 optixGetInstanceInverseTransformFromHandle()

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

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

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

5.1.2.38 optixGetInstanceTransformFromHandle()

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

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

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

5.1.2.39 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)

5.1.2.40 optixGetLaunchDimensions()

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

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

5.1.2.41 optixGetLaunchIndex()

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

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

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

5.1.2.42 optixGetLinearCurveVertexData()

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

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

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

5.1.2.43 optixGetMatrixMotionTransformFromHandle()

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

Returns a pointer to a OptixMatrixMotionTransform from its traversable handle.

Returns 0 if the traversable is not of type OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM.

5.1.2.44 optixGetObjectRayDirection()

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

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

Only available in IS and AH.

5.1.2.45 optixGetObjectRayOrigin()

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

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

Only available in IS and AH.

5.1.2.46 optixGetObjectToWorldTransformMatrix()

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

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

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

5.1.2.47 optixGetPayload_0()

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

Reads the 32-bit payload value at slot 0.

5.1.2.48 optixGetPayload_1()

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

Reads the 32-bit payload value at slot 1.

5.1.2.49 optixGetPayload_10()

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

Reads the 32-bit payload value at slot 10.

5.1.2.50 optixGetPayload_11()

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

Reads the 32-bit payload value at slot 11.

5.1.2.51 optixGetPayload_12()

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

Reads the 32-bit payload value at slot 12.

5.1.2.52 optixGetPayload_13()

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

Reads the 32-bit payload value at slot 13.

5.1.2.53 optixGetPayload_14()

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

Reads the 32-bit payload value at slot 14.

5.1.2.54 optixGetPayload_15()

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

Reads the 32-bit payload value at slot 15.

5.1.2.55 optixGetPayload_16()

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

Reads the 32-bit payload value at slot 16.

5.1.2.56 optixGetPayload_17()

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

Reads the 32-bit payload value at slot 17.

5.1.2.57 optixGetPayload_18()

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

Reads the 32-bit payload value at slot 18.

5.1.2.58 optixGetPayload_19()

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

Reads the 32-bit payload value at slot 19.

5.1.2.59 optixGetPayload_2()

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

Reads the 32-bit payload value at slot 2.

5.1.2.60 optixGetPayload_20()

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

Reads the 32-bit payload value at slot 20.

5.1.2.61 optixGetPayload_21()

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

Reads the 32-bit payload value at slot 21.

5.1.2.62 optixGetPayload_22()

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

Reads the 32-bit payload value at slot 22.

5.1.2.63 optixGetPayload_23()

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

Reads the 32-bit payload value at slot 23.

5.1.2.64 optixGetPayload_24()

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

Reads the 32-bit payload value at slot 24.

5.1.2.65 optixGetPayload_25()

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

Reads the 32-bit payload value at slot 25.

5.1.2.66 optixGetPayload_26()

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

Reads the 32-bit payload value at slot 26.

5.1.2.67 optixGetPayload_27()

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

Reads the 32-bit payload value at slot 27.

5.1.2.68 optixGetPayload_28()

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

Reads the 32-bit payload value at slot 28.

5.1.2.69 optixGetPayload_29()

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

Reads the 32-bit payload value at slot 29.

5.1.2.70 optixGetPayload_3()

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

Reads the 32-bit payload value at slot 3.

5.1.2.71 optixGetPayload_30()

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

Reads the 32-bit payload value at slot 30.

5.1.2.72 optixGetPayload_31()

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

Reads the 32-bit payload value at slot 31.

5.1.2.73 optixGetPayload_4()

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

Reads the 32-bit payload value at slot 4.

5.1.2.74 optixGetPayload_5()

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

Reads the 32-bit payload value at slot 5.

5.1.2.75 optixGetPayload_6()

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

Reads the 32-bit payload value at slot 6.

5.1.2.76 optixGetPayload_7()

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

Reads the 32-bit payload value at slot 7.

5.1.2.77 optixGetPayload_8()

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

Reads the 32-bit payload value at slot 8.

5.1.2.78 optixGetPayload_9()

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

Reads the 32-bit payload value at slot 9.

5.1.2.79 optixGetPrimitiveIndex()

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

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

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

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

5.1.2.80 optixGetPrimitiveType() [1/2]

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

Function interpreting the hit kind associated with the current optixReportIntersection.

5.1.2.81 optixGetPrimitiveType() [2/2]

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

Function interpreting the result of optixGetHitKind().

5.1.2.82 optixGetQuadraticBSplineVertexData()

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

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

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

5.1.2.83 optixGetRayFlags()

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

Returns the rayFlags passed into optixTrace.

Only available in IS, AH, CH, MS

5.1.2.84 optixGetRayTime()

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

Returns the rayTime passed into optixTrace.

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

5.1.2.85 optixGetRayTmax()

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

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

5.1.2.86 optixGetRayTmin()

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

Returns the tmin passed into optixTrace.

Only available in IS, AH, CH, MS

5.1.2.87 optixGetRayVisibilityMask()

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

Returns the visibilityMask passed into optixTrace.

Only available in IS, AH, CH, MS

5.1.2.88 optixGetSbtDataPointer()

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

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

5.1.2.89 optixGetSbtGASIndex()

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

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

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

5.1.2.90 optixGetSphereData()

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

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

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

5.1.2.91 optixGetSRTMotionTransformFromHandle()

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

Returns a pointer to a OptixSRTMotionTransform from its traversable handle.

Returns 0 if the traversable is not of type OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM.

5.1.2.92 optixGetStaticTransformFromHandle()

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

Returns a pointer to a OptixStaticTransform from its traversable handle.

Returns 0 if the traversable is not of type OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM.

5.1.2.93 optixGetTransformListHandle()

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

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

Only available in IS, AH, CH, EX

5.1.2.94 optixGetTransformListSize()

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

Returns the number of transforms on the current transform list.

Only available in IS, AH, CH, EX

5.1.2.95 optixGetTransformTypeFromHandle()

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

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

5.1.2.96 optixGetTriangleBarycentrics()

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

Convenience function that returns the first two attributes as floats.

When using OptixBuildInputTriangleArray objects, during intersection the barycentric coordinates are stored into the first two attribute registers.

5.1.2.97 optixGetTriangleVertexData()

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

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

If motion is disabled via OptixPipelineCompileOptions::usesMotionBlur, or the GAS does not contain motion, the time parameter is ignored.

5.1.2.98 optixGetWorldRayDirection()

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

Returns the rayDirection passed into optixTrace.

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

5.1.2.99 optixGetWorldRayOrigin()

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

Returns the rayOrigin passed into optixTrace.

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

5.1.2.100 optixGetWorldToObjectTransformMatrix()

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

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

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

5.1.2.101 optixIgnoreIntersection()

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

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

Available only in AH.

5.1.2.102 optixIsBackFaceHit() [1/2]

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

Function interpreting the hit kind associated with the current optixReportIntersection.

5.1.2.103 optixIsBackFaceHit() [2/2]

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

Function interpreting the result of optixGetHitKind().

5.1.2.104 optixIsFrontFaceHit() [1/2]

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

Function interpreting the hit kind associated with the current optixReportIntersection.

5.1.2.105 optixIsFrontFaceHit() [2/2]

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

Function interpreting the result of optixGetHitKind().

5.1.2.106 optixIsTriangleBackFaceHit()

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

Convenience function interpreting the result of optixGetHitKind( ).

5.1.2.107 optixIsTriangleFrontFaceHit( )

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

Convenience function interpreting the result of optixGetHitKind( ).

5.1.2.108 optixIsTriangleHit( )

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

Convenience function interpreting the result of optixGetHitKind( ).

5.1.2.109 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	*hitT*
in	*hitKind*

5.1.2.110 optixReportIntersection( ) [2/9]

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

Reports an intersection (overload with 1 attribute register).

See also optixReportIntersection(float,unsigned int)

5.1.2.111 optixReportIntersection() [3/9]

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

Reports an intersection (overload with 2 attribute registers).

See also optixReportIntersection(float,unsigned int)

5.1.2.112 optixReportIntersection() [4/9]

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

Reports an intersection (overload with 3 attribute registers).

See also optixReportIntersection(float,unsigned int)

5.1.2.113 optixReportIntersection() [5/9]

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

Reports an intersection (overload with 4 attribute registers).

See also optixReportIntersection(float,unsigned int)

5.1.2.114 optixReportIntersection() [6/9]

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

Reports an intersection (overload with 5 attribute registers).

See also optixReportIntersection(float,unsigned int)

5.1.2.115 optixReportIntersection() [7/9]

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

Reports an intersection (overload with 6 attribute registers).

See also optixReportIntersection(float,unsigned int)

5.1.2.116 optixReportIntersection() [8/9]

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

Reports an intersection (overload with 7 attribute registers).

See also optixReportIntersection(float,unsigned int)

5.1.2.117 optixReportIntersection() [9/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
          float hitT,
          unsigned int hitKind,
          unsigned int a0,
          unsigned int a1,
          unsigned int a2,
          unsigned int a3,
          unsigned int a4,
          unsigned int a5,
          unsigned int a6,
          unsigned int a7 )  [static]
```

Reports an intersection (overload with 8 attribute registers).

See also optixReportIntersection(float,unsigned int)

5.1.2.118 optixSetPayload_0()

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

Writes the 32-bit payload value at slot 0.

5.1.2.119 optixSetPayload_1()

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

Writes the 32-bit payload value at slot 1.

5.1.2.120 optixSetPayload_10()

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

Writes the 32-bit payload value at slot 10.

5.1.2.121 optixSetPayload_11()

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

Writes the 32-bit payload value at slot 11.

5.1.2.122 optixSetPayload_12()

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

Writes the 32-bit payload value at slot 12.

5.1.2.123 optixSetPayload_13()

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

Writes the 32-bit payload value at slot 13.

5.1.2.124 optixSetPayload_14()

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

Writes the 32-bit payload value at slot 14.

5.1.2.125 optixSetPayload_15()

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

Writes the 32-bit payload value at slot 15.

5.1.2.126 optixSetPayload_16()

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

Writes the 32-bit payload value at slot 16.

5.1.2.127 optixSetPayload_17()

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

Writes the 32-bit payload value at slot 17.

5.1.2.128 optixSetPayload_18()

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

Writes the 32-bit payload value at slot 18.

5.1.2.129 optixSetPayload_19()

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

Writes the 32-bit payload value at slot 19.

5.1.2.130 optixSetPayload_2()

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

Writes the 32-bit payload value at slot 2.

5.1.2.131 optixSetPayload_20()

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

Writes the 32-bit payload value at slot 20.

5.1.2.132 optixSetPayload_21()

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

Writes the 32-bit payload value at slot 21.

5.1.2.133 optixSetPayload_22()

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

Writes the 32-bit payload value at slot 22.

5.1.2.134 optixSetPayload_23()

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

Writes the 32-bit payload value at slot 23.

5.1.2.135 optixSetPayload_24()

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

Writes the 32-bit payload value at slot 24.

5.1.2.136 optixSetPayload_25()

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

Writes the 32-bit payload value at slot 25.

5.1.2.137 optixSetPayload_26()

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

Writes the 32-bit payload value at slot 26.

5.1.2.138 optixSetPayload_27()

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

Writes the 32-bit payload value at slot 27.

5.1.2.139 optixSetPayload_28()

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

Writes the 32-bit payload value at slot 28.

5.1.2.140 optixSetPayload_29()

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

Writes the 32-bit payload value at slot 29.

5.1.2.141 optixSetPayload_3()

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

Writes the 32-bit payload value at slot 3.

5.1.2.142 optixSetPayload_30()

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

Writes the 32-bit payload value at slot 30.

5.1.2.143 optixSetPayload_31()

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

Writes the 32-bit payload value at slot 31.

5.1.2.144 optixSetPayload_4()

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

Writes the 32-bit payload value at slot 4.

5.1.2.145 optixSetPayload_5()

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

Writes the 32-bit payload value at slot 5.

5.1.2.146 optixSetPayload_6()

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

Writes the 32-bit payload value at slot 6.

5.1.2.147 optixSetPayload_7()

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

Writes the 32-bit payload value at slot 7.

5.1.2.148 optixSetPayload_8()

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

Writes the 32-bit payload value at slot 8.

5.1.2.149 optixSetPayload_9()

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

Writes the 32-bit payload value at slot 9.

5.1.2.150 optixSetPayloadTypes()

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

Specify the supported payload types for a program.

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

5.1.2.151 optixTerminateRay()

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

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

Available only in AH.

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

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

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

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

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

The granularity indicates the size of texel groups that are represented by an 8x8 bitmask. For example,

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

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

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

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

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

5.1.2.153 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	*tex*	CUDA texture object (cast to 64-bit integer)
in	*texInfo*	Texture info packed into 32-bit integer, described below.
in	*x*	Texture coordinate
in	*y*	Texture coordinate
in	*dPdx_x*	Derivative of x coordinte, which determines level of detail.
in	*dPdx_y*	Derivative of x coordinte, which determines level of detail.
in	*dPdy_x*	Derivative of y coordinte, which determines level of detail.
in	*dPdy_y*	Derivative of y coordinte, which determines level of detail.
in	*coarse*	Requests footprint from coarse miplevel, when the footprint spans two levels.
out	*singleMipLevel*	Result indicating whether the footprint spans only a single miplevel.

See also optixTexFootprint2D(unsigned long long,unsigned int,float,float,unsigned int∗)

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

See also optixTexFootprint2D(unsigned long long,unsigned int,float,float,unsigned int*)

5.1.2.155 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	*exceptionCode*	The exception code to be thrown.

5.1.2.156 optixThrowException() [2/9]

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

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

See also optixThrowException(int)

5.1.2.157 optixThrowException() [3/9]

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

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

See also optixThrowException(int)

5.1.2.158 optixThrowException() [4/9]

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

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

See also optixThrowException(int)

5.1.2.159 optixThrowException() [5/9]

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

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

See also optixThrowException(int)

5.1.2.160 optixThrowException() [6/9]

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

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

See also optixThrowException(int)

5.1.2.161 optixThrowException() [7/9]

```
static __forceinline__ __device__ void optixThrowException (
          int exceptionCode,
```

```
            unsigned int exceptionDetail0,
            unsigned int exceptionDetail1,
            unsigned int exceptionDetail2,
            unsigned int exceptionDetail3,
            unsigned int exceptionDetail4,
            unsigned int exceptionDetail5 )  [static]
```

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

See also optixThrowException(int)

5.1.2.162 optixThrowException() [8/9]

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

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

See also optixThrowException(int)

5.1.2.163 optixThrowException() [9/9]

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

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

See also optixThrowException(int)

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

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

5.1.2.166 optixTransformNormalFromObjectToWorldSpace()

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

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

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

5.1.2.167 optixTransformNormalFromWorldToObjectSpace()

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

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

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

5.1.2.168 optixTransformPointFromObjectToWorldSpace()

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

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

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

5.1.2.169 optixTransformPointFromWorldToObjectSpace()

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

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

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

5.1.2.170 optixTransformVectorFromObjectToWorldSpace()

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

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

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

5.1.2.171 optixTransformVectorFromWorldToObjectSpace()

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

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

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

5.1.2.172 optixUndefinedValue()

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

Returns an undefined value.

5.2 Host API

Modules

- Error handling
- Device context
- Pipelines
- Modules
- Tasks
- Program groups
- Launches
- Acceleration structures
- Denoiser

5.2.1 Detailed Description

OptiX Host API.

5.3 Error handling

5.4 Device context

5.5 Pipelines

5.6 Modules

5.7 Tasks

5.8 Program groups

5.9 Launches

5.10 Acceleration structures

5.11 Denoiser

5.12 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 OptixImage2D

- struct OptixDenoiserOptions
- struct OptixDenoiserGuideLayer
- struct OptixDenoiserLayer
- struct OptixDenoiserParams
- struct OptixDenoiserSizes
- struct OptixModuleCompileBoundValueEntry
- struct OptixPayloadType
- struct OptixModuleCompileOptions
- struct OptixProgramGroupSingleModule
- struct OptixProgramGroupHitgroup
- struct OptixProgramGroupCallables
- struct OptixProgramGroupDesc
- struct OptixProgramGroupOptions
- struct OptixPipelineCompileOptions
- struct OptixPipelineLinkOptions
- struct OptixShaderBindingTable
- struct OptixStackSizes
- struct OptixBuiltinISOptions

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_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
- #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT (0)
- #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE (1)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT (2)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE (3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT (-1)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE (-2)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT (-3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE (-4)
- #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
- #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12

Typedefs

- typedef unsigned long long CUdeviceptr
- typedef struct OptixDeviceContext_t * OptixDeviceContext
- typedef struct OptixModule_t * OptixModule
- typedef struct OptixProgramGroup_t * OptixProgramGroup
- typedef struct OptixPipeline_t * OptixPipeline

- typedef struct OptixDenoiser_t * OptixDenoiser
- typedef struct OptixTask_t * OptixTask
- typedef unsigned long long OptixTraversableHandle
- typedef unsigned int OptixVisibilityMask
- typedef enum OptixResult OptixResult
- typedef enum OptixDeviceProperty OptixDeviceProperty
- typedef void(* OptixLogCallback) (unsigned int level, const char *tag, const char *message, void *cbdata)
- typedef enum OptixDeviceContextValidationMode OptixDeviceContextValidationMode
- typedef struct OptixDeviceContextOptions OptixDeviceContextOptions
- typedef enum OptixGeometryFlags OptixGeometryFlags
- typedef enum OptixHitKind OptixHitKind
- typedef enum OptixIndicesFormat OptixIndicesFormat
- typedef enum OptixVertexFormat OptixVertexFormat
- typedef enum OptixTransformFormat OptixTransformFormat
- typedef enum 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 OptixPixelFormat OptixPixelFormat
- typedef struct OptixImage2D OptixImage2D
- typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
- typedef struct OptixDenoiserOptions OptixDenoiserOptions
- typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
- typedef struct OptixDenoiserLayer OptixDenoiserLayer
- typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
- typedef struct OptixDenoiserParams OptixDenoiserParams
- typedef struct OptixDenoiserSizes OptixDenoiserSizes
- typedef enum OptixRayFlags OptixRayFlags
- typedef enum OptixTransformType OptixTransformType
- typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags
- typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
- typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
- typedef enum OptixModuleCompileState OptixModuleCompileState
- typedef struct OptixModuleCompileBoundValueEntry OptixModuleCompileBoundValueEntry
- typedef enum OptixPayloadTypeID OptixPayloadTypeID
- typedef enum OptixPayloadSemantics OptixPayloadSemantics
- typedef struct OptixPayloadType OptixPayloadType
- typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
- typedef enum OptixProgramGroupKind OptixProgramGroupKind
- typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
- typedef struct OptixProgramGroupSingleModule OptixProgramGroupSingleModule
- typedef struct OptixProgramGroupHitgroup OptixProgramGroupHitgroup
- typedef struct OptixProgramGroupCallables OptixProgramGroupCallables
- typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
- typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
- typedef enum OptixExceptionCodes OptixExceptionCodes
- typedef enum OptixExceptionFlags OptixExceptionFlags
- typedef struct OptixPipelineCompileOptions OptixPipelineCompileOptions
- typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
- typedef struct OptixShaderBindingTable OptixShaderBindingTable
- typedef struct OptixStackSizes OptixStackSizes
- typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
- typedef OptixResult() OptixQueryFunctionTable_t(int abiId, unsigned int numOptions, OptixQueryFunctionTableOptions *, const void **, void *functionTable, size_t sizeOfTable)
- typedef struct OptixBuiltinISOptions OptixBuiltinISOptions

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_PTX = 7200 ,
  OPTIX_ERROR_INVALID_LAUNCH_PARAMETER = 7201 ,
  OPTIX_ERROR_INVALID_PAYLOAD_ACCESS = 7202 ,
  OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS = 7203 ,
  OPTIX_ERROR_INVALID_FUNCTION_USE = 7204 ,
  OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS = 7205 ,
  OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250 ,
  OPTIX_ERROR_PIPELINE_LINK_ERROR = 7251 ,
  OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE = 7270 ,
  OPTIX_ERROR_INTERNAL_COMPILER_ERROR = 7299 ,
  OPTIX_ERROR_DENOISER_MODEL_NOT_SET = 7300 ,
  OPTIX_ERROR_DENOISER_NOT_INITIALIZED = 7301 ,
  OPTIX_ERROR_NOT_COMPATIBLE = 7400 ,
  OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH = 7500 ,
  OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED = 7501 ,
  OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID = 7502 ,
  OPTIX_ERROR_NOT_SUPPORTED = 7800 ,
  OPTIX_ERROR_UNSUPPORTED_ABI_VERSION = 7801 ,
  OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH = 7802 ,
  OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS = 7803 ,
  OPTIX_ERROR_LIBRARY_NOT_FOUND = 7804 ,
  OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND = 7805 ,
  OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE = 7806 ,
  OPTIX_ERROR_DEVICE_OUT_OF_MEMORY = 7807 ,
  OPTIX_ERROR_CUDA_ERROR = 7900 ,
  OPTIX_ERROR_INTERNAL_ERROR = 7990 ,
  OPTIX_ERROR_UNKNOWN = 7999 }
- enum OptixDeviceProperty {
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004 ,
  OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009 }

- enum OptixDeviceContextValidationMode {
  OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0 ,
  OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF }
- enum OptixGeometryFlags {
  OPTIX_GEOMETRY_FLAG_NONE = 0 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
  OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u << 1 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 2 }
- enum OptixHitKind {
  OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE ,
  OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF }
- enum OptixIndicesFormat {
  OPTIX_INDICES_FORMAT_NONE = 0 ,
  OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102 ,
  OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103 }
- enum OptixVertexFormat {
  OPTIX_VERTEX_FORMAT_NONE = 0 ,
  OPTIX_VERTEX_FORMAT_FLOAT3 = 0x2121 ,
  OPTIX_VERTEX_FORMAT_FLOAT2 = 0x2122 ,
  OPTIX_VERTEX_FORMAT_HALF3 = 0x2123 ,
  OPTIX_VERTEX_FORMAT_HALF2 = 0x2124 ,
  OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125 ,
  OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126 }
- enum OptixTransformFormat {
  OPTIX_TRANSFORM_FORMAT_NONE = 0 ,
  OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1 }
- enum 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_SPHERE = 0x2506 ,
  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_SPHERE = 1 << 6 ,
  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 OptixPixelFormat {
  OPTIX_PIXEL_FORMAT_HALF2 = 0x2207 ,
  OPTIX_PIXEL_FORMAT_HALF3 = 0x2201 ,
  OPTIX_PIXEL_FORMAT_HALF4 = 0x2202 ,
  OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208 ,
  OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203 ,
  OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204 ,
  OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205 ,
  OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206 ,
  OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209 }

- enum OptixDenoiserModelKind {
  OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322 ,
  OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323 ,
  OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326 ,
  OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328 }
- enum OptixDenoiserAlphaMode {
  OPTIX_DENOISER_ALPHA_MODE_COPY = 0 ,
  OPTIX_DENOISER_ALPHA_MODE_ALPHA_AS_AOV = 1 ,
  OPTIX_DENOISER_ALPHA_MODE_FULL_DENOISE_PASS = 2 }
- enum OptixRayFlags {
  OPTIX_RAY_FLAG_NONE = 0u ,
  OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
  OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u << 1 ,
  OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u << 2 ,
  OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u << 3 ,
  OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u << 4 ,
  OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u << 5 ,
  OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u << 6 ,
  OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u << 7 ,
  OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 10 }
- enum OptixTransformType {
  OPTIX_TRANSFORM_TYPE_NONE = 0 ,
  OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM = 1 ,
  OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM = 2 ,
  OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM = 3 ,
  OPTIX_TRANSFORM_TYPE_INSTANCE = 4 }
- enum OptixTraversableGraphFlags {
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u << 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u << 1 }
- enum OptixCompileOptimizationLevel {
  OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343 }
- enum OptixCompileDebugLevel {
  OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT = 0 ,
  OPTIX_COMPILE_DEBUG_LEVEL_NONE = 0x2350 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL = 0x2351 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353 ,
  OPTIX_COMPILE_DEBUG_LEVEL_FULL = 0x2352 }
- enum OptixModuleCompileState {
  OPTIX_MODULE_COMPILE_STATE_NOT_STARTED = 0x2360 ,
  OPTIX_MODULE_COMPILE_STATE_STARTED = 0x2361 ,
  OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE = 0x2362 ,
  OPTIX_MODULE_COMPILE_STATE_FAILED = 0x2363 ,
  OPTIX_MODULE_COMPILE_STATE_COMPLETED = 0x2364 }
- enum OptixPayloadTypeID {
  OPTIX_PAYLOAD_TYPE_DEFAULT = 0 ,

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

- enum OptixPayloadSemantics {
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ = 1u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE = 2u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_READ = 1u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_WRITE = 2u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE = 3u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_READ = 1u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_WRITE = 2u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE = 3u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_READ = 1u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_WRITE = 2u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE = 3u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_READ = 1u << 8 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_WRITE = 2u << 8 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE = 3u << 8 }
- enum OptixProgramGroupKind {
  OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421 ,
  OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422 ,
  OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423 ,
  OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424 ,
  OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425 }
- enum OptixProgramGroupFlags { OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0 }
- enum OptixExceptionCodes {
  OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1 ,
  OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED = -3 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE = -5 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT = -6 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT = -7 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE = -8 ,
  OPTIX_EXCEPTION_CODE_INVALID_RAY = -9 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH = -10 ,
  OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH = -11 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT = -12 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD = -13 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD = -14 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS = -15 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0 = -16 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_1 = -17 ,

OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2 = -18 ,
OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS = -32 ,
OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH = -33 }

- enum OptixExceptionFlags {
OPTIX_EXCEPTION_FLAG_NONE = 0 ,
OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u << 0 ,
OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u << 1 ,
OPTIX_EXCEPTION_FLAG_USER = 1u << 2 ,
OPTIX_EXCEPTION_FLAG_DEBUG = 1u << 3 }
- enum OptixQueryFunctionTableOptions { OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0 }

Variables

- OptixLogCallback OptixDeviceContextOptions::logCallbackFunction
- void * OptixDeviceContextOptions::logCallbackData
- int OptixDeviceContextOptions::logCallbackLevel
- OptixDeviceContextValidationMode OptixDeviceContextOptions::validationMode
- unsigned int OptixOpacityMicromapUsageCount::count
- unsigned int OptixOpacityMicromapUsageCount::subdivisionLevel
- OptixOpacityMicromapFormat OptixOpacityMicromapUsageCount::format
- OptixOpacityMicromapArrayIndexingMode OptixBuildInputOpacityMicromap::indexingMode
- CUdeviceptr OptixBuildInputOpacityMicromap::opacityMicromapArray
- CUdeviceptr OptixBuildInputOpacityMicromap::indexBuffer
- unsigned int OptixBuildInputOpacityMicromap::indexSizeInBytes
- unsigned int OptixBuildInputOpacityMicromap::indexStrideInBytes
- unsigned int OptixBuildInputOpacityMicromap::indexOffset
- unsigned int OptixBuildInputOpacityMicromap::numMicromapUsageCounts
- const OptixOpacityMicromapUsageCount * OptixBuildInputOpacityMicromap::micromapUsageCounts
- CUdeviceptr OptixRelocateInputOpacityMicromap::opacityMicromapArray
- const CUdeviceptr * OptixBuildInputTriangleArray::vertexBuffers
- unsigned int OptixBuildInputTriangleArray::numVertices
- OptixVertexFormat OptixBuildInputTriangleArray::vertexFormat
- unsigned int OptixBuildInputTriangleArray::vertexStrideInBytes
- CUdeviceptr OptixBuildInputTriangleArray::indexBuffer
- unsigned int OptixBuildInputTriangleArray::numIndexTriplets
- OptixIndicesFormat OptixBuildInputTriangleArray::indexFormat
- unsigned int OptixBuildInputTriangleArray::indexStrideInBytes
- CUdeviceptr OptixBuildInputTriangleArray::preTransform
- const unsigned int * OptixBuildInputTriangleArray::flags
- unsigned int OptixBuildInputTriangleArray::numSbtRecords
- CUdeviceptr OptixBuildInputTriangleArray::sbtIndexOffsetBuffer
- unsigned int OptixBuildInputTriangleArray::sbtIndexOffsetSizeInBytes
- unsigned int OptixBuildInputTriangleArray::sbtIndexOffsetStrideInBytes
- unsigned int OptixBuildInputTriangleArray::primitiveIndexOffset
- OptixTransformFormat OptixBuildInputTriangleArray::transformFormat
- OptixBuildInputOpacityMicromap OptixBuildInputTriangleArray::opacityMicromap
- unsigned int OptixRelocateInputTriangleArray::numSbtRecords
- OptixRelocateInputOpacityMicromap OptixRelocateInputTriangleArray::opacityMicromap
- OptixPrimitiveType OptixBuildInputCurveArray::curveType

- unsigned int OptixBuildInputCurveArray::numPrimitives
- const CUdeviceptr * OptixBuildInputCurveArray::vertexBuffers
- unsigned int OptixBuildInputCurveArray::numVertices
- unsigned int OptixBuildInputCurveArray::vertexStrideInBytes
- const CUdeviceptr * OptixBuildInputCurveArray::widthBuffers
- unsigned int OptixBuildInputCurveArray::widthStrideInBytes
- const CUdeviceptr * OptixBuildInputCurveArray::normalBuffers
- unsigned int OptixBuildInputCurveArray::normalStrideInBytes
- CUdeviceptr OptixBuildInputCurveArray::indexBuffer
- unsigned int OptixBuildInputCurveArray::indexStrideInBytes
- unsigned int OptixBuildInputCurveArray::flag
- unsigned int OptixBuildInputCurveArray::primitiveIndexOffset
- unsigned int OptixBuildInputCurveArray::endcapFlags
- const CUdeviceptr * OptixBuildInputSphereArray::vertexBuffers
- unsigned int OptixBuildInputSphereArray::vertexStrideInBytes
- unsigned int OptixBuildInputSphereArray::numVertices
- const CUdeviceptr * OptixBuildInputSphereArray::radiusBuffers
- unsigned int OptixBuildInputSphereArray::radiusStrideInBytes
- int OptixBuildInputSphereArray::singleRadius
- const unsigned int * OptixBuildInputSphereArray::flags
- unsigned int OptixBuildInputSphereArray::numSbtRecords
- CUdeviceptr OptixBuildInputSphereArray::sbtIndexOffsetBuffer
- unsigned int OptixBuildInputSphereArray::sbtIndexOffsetSizeInBytes
- unsigned int OptixBuildInputSphereArray::sbtIndexOffsetStrideInBytes
- unsigned int OptixBuildInputSphereArray::primitiveIndexOffset
- float OptixAabb::minX
- float OptixAabb::minY
- float OptixAabb::minZ
- float OptixAabb::maxX
- float OptixAabb::maxY
- float OptixAabb::maxZ
- const CUdeviceptr * OptixBuildInputCustomPrimitiveArray::aabbBuffers
- unsigned int OptixBuildInputCustomPrimitiveArray::numPrimitives
- unsigned int OptixBuildInputCustomPrimitiveArray::strideInBytes
- const unsigned int * OptixBuildInputCustomPrimitiveArray::flags
- unsigned int OptixBuildInputCustomPrimitiveArray::numSbtRecords
- CUdeviceptr OptixBuildInputCustomPrimitiveArray::sbtIndexOffsetBuffer
- unsigned int OptixBuildInputCustomPrimitiveArray::sbtIndexOffsetSizeInBytes
- unsigned int OptixBuildInputCustomPrimitiveArray::sbtIndexOffsetStrideInBytes
- unsigned int OptixBuildInputCustomPrimitiveArray::primitiveIndexOffset
- CUdeviceptr OptixBuildInputInstanceArray::instances
- unsigned int OptixBuildInputInstanceArray::numInstances
- unsigned int OptixBuildInputInstanceArray::instanceStride
- unsigned int OptixRelocateInputInstanceArray::numInstances
- CUdeviceptr OptixRelocateInputInstanceArray::traversableHandles
- OptixBuildInputType OptixBuildInput::type
- OptixBuildInputTriangleArray OptixBuildInput::triangleArray
- OptixBuildInputCurveArray OptixBuildInput::curveArray
- OptixBuildInputSphereArray OptixBuildInput::sphereArray
- OptixBuildInputCustomPrimitiveArray OptixBuildInput::customPrimitiveArray

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

- OptixBuildInputType OptixRelocateInput::type
- OptixRelocateInputInstanceArray OptixRelocateInput::instanceArray
- OptixRelocateInputTriangleArray OptixRelocateInput::triangleArray
- union {
  OptixRelocateInputInstanceArray OptixRelocateInput::instanceArray
  OptixRelocateInputTriangleArray OptixRelocateInput::triangleArray
  };

- float OptixInstance::transform [12]
- unsigned int OptixInstance::instanceId
- unsigned int OptixInstance::sbtOffset
- unsigned int OptixInstance::visibilityMask
- unsigned int OptixInstance::flags
- OptixTraversableHandle OptixInstance::traversableHandle
- unsigned int OptixInstance::pad [2]
- unsigned int OptixOpacityMicromapDesc::byteOffset
- unsigned short OptixOpacityMicromapDesc::subdivisionLevel
- unsigned short OptixOpacityMicromapDesc::format
- unsigned int OptixOpacityMicromapHistogramEntry::count
- unsigned int OptixOpacityMicromapHistogramEntry::subdivisionLevel
- OptixOpacityMicromapFormat OptixOpacityMicromapHistogramEntry::format
- OptixOpacityMicromapFlags OptixOpacityMicromapArrayBuildInput::flags
- CUdeviceptr OptixOpacityMicromapArrayBuildInput::inputBuffer
- CUdeviceptr OptixOpacityMicromapArrayBuildInput::perMicromapDescBuffer
- unsigned int OptixOpacityMicromapArrayBuildInput::perMicromapDescStrideInBytes
- unsigned int OptixOpacityMicromapArrayBuildInput::numMicromapHistogramEntries
- const OptixOpacityMicromapHistogramEntry * OptixOpacityMicromapArrayBuildInput ::micromapHistogramEntries
- size_t OptixMicromapBufferSizes::outputSizeInBytes
- size_t OptixMicromapBufferSizes::tempSizeInBytes
- CUdeviceptr OptixMicromapBuffers::output
- size_t OptixMicromapBuffers::outputSizeInBytes
- CUdeviceptr OptixMicromapBuffers::temp
- size_t OptixMicromapBuffers::tempSizeInBytes
- unsigned short OptixMotionOptions::numKeys
- unsigned short OptixMotionOptions::flags
- float OptixMotionOptions::timeBegin
- float OptixMotionOptions::timeEnd
- unsigned int OptixAccelBuildOptions::buildFlags
- OptixBuildOperation OptixAccelBuildOptions::operation

- OptixMotionOptions OptixAccelBuildOptions::motionOptions
- size_t OptixAccelBufferSizes::outputSizeInBytes
- size_t OptixAccelBufferSizes::tempSizeInBytes
- size_t OptixAccelBufferSizes::tempUpdateSizeInBytes
- CUdeviceptr OptixAccelEmitDesc::result
- OptixAccelPropertyType OptixAccelEmitDesc::type
- unsigned long long OptixRelocationInfo::info [4]
- OptixTraversableHandle OptixStaticTransform::child
- unsigned int OptixStaticTransform::pad [2]
- float OptixStaticTransform::transform [12]
- float OptixStaticTransform::invTransform [12]
- OptixTraversableHandle OptixMatrixMotionTransform::child
- OptixMotionOptions OptixMatrixMotionTransform::motionOptions
- unsigned int OptixMatrixMotionTransform::pad [3]
- float OptixMatrixMotionTransform::transform [2][12]
- OptixTraversableHandle OptixSRTMotionTransform::child
- OptixMotionOptions OptixSRTMotionTransform::motionOptions
- unsigned int OptixSRTMotionTransform::pad [3]
- OptixSRTData OptixSRTMotionTransform::srtData [2]
- CUdeviceptr OptixImage2D::data
- unsigned int OptixImage2D::width
- unsigned int OptixImage2D::height
- unsigned int OptixImage2D::rowStrideInBytes
- unsigned int OptixImage2D::pixelStrideInBytes
- OptixPixelFormat OptixImage2D::format
- unsigned int OptixDenoiserOptions::guideAlbedo
- unsigned int OptixDenoiserOptions::guideNormal
- OptixImage2D OptixDenoiserGuideLayer::albedo
- OptixImage2D OptixDenoiserGuideLayer::normal
- OptixImage2D OptixDenoiserGuideLayer::flow
- OptixImage2D OptixDenoiserGuideLayer::previousOutputInternalGuideLayer
- OptixImage2D OptixDenoiserGuideLayer::outputInternalGuideLayer
- OptixImage2D OptixDenoiserLayer::input
- OptixImage2D OptixDenoiserLayer::previousOutput
- OptixImage2D OptixDenoiserLayer::output
- OptixDenoiserAlphaMode OptixDenoiserParams::denoiseAlpha
- CUdeviceptr OptixDenoiserParams::hdrIntensity
- float OptixDenoiserParams::blendFactor
- CUdeviceptr OptixDenoiserParams::hdrAverageColor
- unsigned int OptixDenoiserParams::temporalModeUsePreviousLayers
- size_t OptixDenoiserSizes::stateSizeInBytes
- size_t OptixDenoiserSizes::withOverlapScratchSizeInBytes
- size_t OptixDenoiserSizes::withoutOverlapScratchSizeInBytes
- unsigned int OptixDenoiserSizes::overlapWindowSizeInPixels
- size_t OptixDenoiserSizes::computeAverageColorSizeInBytes
- size_t OptixDenoiserSizes::computeIntensitySizeInBytes
- size_t OptixDenoiserSizes::internalGuideLayerPixelSizeInBytes
- size_t OptixModuleCompileBoundValueEntry::pipelineParamOffsetInBytes
- size_t OptixModuleCompileBoundValueEntry::sizeInBytes
- const void * OptixModuleCompileBoundValueEntry::boundValuePtr

- const char * OptixModuleCompileBoundValueEntry::annotation
- unsigned int OptixPayloadType::numPayloadValues
- const unsigned int * OptixPayloadType::payloadSemantics
- int OptixModuleCompileOptions::maxRegisterCount
- OptixCompileOptimizationLevel OptixModuleCompileOptions::optLevel
- OptixCompileDebugLevel OptixModuleCompileOptions::debugLevel
- const OptixModuleCompileBoundValueEntry * OptixModuleCompileOptions::boundValues
- unsigned int OptixModuleCompileOptions::numBoundValues
- unsigned int OptixModuleCompileOptions::numPayloadTypes
- OptixPayloadType * OptixModuleCompileOptions::payloadTypes
- OptixModule OptixProgramGroupSingleModule::module
- const char * OptixProgramGroupSingleModule::entryFunctionName
- OptixModule OptixProgramGroupHitgroup::moduleCH
- const char * OptixProgramGroupHitgroup::entryFunctionNameCH
- OptixModule OptixProgramGroupHitgroup::moduleAH
- const char * OptixProgramGroupHitgroup::entryFunctionNameAH
- OptixModule OptixProgramGroupHitgroup::moduleIS
- const char * OptixProgramGroupHitgroup::entryFunctionNameIS
- OptixModule OptixProgramGroupCallables::moduleDC
- const char * OptixProgramGroupCallables::entryFunctionNameDC
- OptixModule OptixProgramGroupCallables::moduleCC
- const char * OptixProgramGroupCallables::entryFunctionNameCC
- OptixProgramGroupKind OptixProgramGroupDesc::kind
- unsigned int OptixProgramGroupDesc::flags
- OptixProgramGroupSingleModule OptixProgramGroupDesc::raygen
- OptixProgramGroupSingleModule OptixProgramGroupDesc::miss
- OptixProgramGroupSingleModule OptixProgramGroupDesc::exception
- OptixProgramGroupCallables OptixProgramGroupDesc::callables
- OptixProgramGroupHitgroup OptixProgramGroupDesc::hitgroup
- union {
  OptixProgramGroupSingleModule OptixProgramGroupDesc::raygen
  OptixProgramGroupSingleModule OptixProgramGroupDesc::miss
  OptixProgramGroupSingleModule OptixProgramGroupDesc::exception
  OptixProgramGroupCallables OptixProgramGroupDesc::callables
  OptixProgramGroupHitgroup OptixProgramGroupDesc::hitgroup
  };

- OptixPayloadType * OptixProgramGroupOptions::payloadType
- int OptixPipelineCompileOptions::usesMotionBlur
- unsigned int OptixPipelineCompileOptions::traversableGraphFlags
- int OptixPipelineCompileOptions::numPayloadValues
- int OptixPipelineCompileOptions::numAttributeValues
- unsigned int OptixPipelineCompileOptions::exceptionFlags
- const char * OptixPipelineCompileOptions::pipelineLaunchParamsVariableName
- unsigned int OptixPipelineCompileOptions::usesPrimitiveTypeFlags
- int OptixPipelineCompileOptions::allowOpacityMicromaps
- unsigned int OptixPipelineLinkOptions::maxTraceDepth
- OptixCompileDebugLevel OptixPipelineLinkOptions::debugLevel
- CUdeviceptr OptixShaderBindingTable::raygenRecord
- CUdeviceptr OptixShaderBindingTable::exceptionRecord

- unsigned int OptixStackSizes::cssRG
- unsigned int OptixStackSizes::cssMS
- unsigned int OptixStackSizes::cssCH
- unsigned int OptixStackSizes::cssAH
- unsigned int OptixStackSizes::cssIS
- unsigned int OptixStackSizes::cssCC
- unsigned int OptixStackSizes::dssDC
- OptixPrimitiveType OptixBuiltinISOptions::builtinISModuleType
- int OptixBuiltinISOptions::usesMotionBlur
- unsigned int OptixBuiltinISOptions::buildFlags
- unsigned int OptixBuiltinISOptions::curveEndcapFlags

Parameters describing the SRT transformation

- float OptixSRTData::sx
- float OptixSRTData::a
- float OptixSRTData::b
- float OptixSRTData::pvx
- float OptixSRTData::sy
- float OptixSRTData::c
- float OptixSRTData::pvy
- float OptixSRTData::sz
- float OptixSRTData::pvz
- float OptixSRTData::qx
- float OptixSRTData::qy
- float OptixSRTData::qz
- float OptixSRTData::qw
- float OptixSRTData::tx
- float OptixSRTData::ty
- float OptixSRTData::tz

- CUdeviceptr OptixShaderBindingTable::missRecordBase
- unsigned int OptixShaderBindingTable::missRecordStrideInBytes
- unsigned int OptixShaderBindingTable::missRecordCount

- CUdeviceptr OptixShaderBindingTable::hitgroupRecordBase
- unsigned int OptixShaderBindingTable::hitgroupRecordStrideInBytes
- unsigned int OptixShaderBindingTable::hitgroupRecordCount

- CUdeviceptr OptixShaderBindingTable::callablesRecordBase
- unsigned int OptixShaderBindingTable::callablesRecordStrideInBytes
- unsigned int OptixShaderBindingTable::callablesRecordCount

5.12.1 Detailed Description

OptiX Types.

5.12.2 Macro Definition Documentation

5.12.2.1 OPTIX_AABB_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8ull
```

Alignment requirement for OptixBuildInputCustomPrimitiveArray::aabbBuffers.

5.12.2.2 OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128ull
```

Alignment requirement for output and temporay buffers for acceleration structures.

5.12.2.3 OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
```

Maximum number of payload types allowed.

5.12.2.4 OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
```

Maximum number of payload values allowed.

5.12.2.5 OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
```

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

5.12.2.6 OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT

```
#define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16ull
```

Alignment requirement for OptixBuildInputTriangleArray::preTransform.

5.12.2.7 OPTIX_INSTANCE_BYTE_ALIGNMENT

```
#define OPTIX_INSTANCE_BYTE_ALIGNMENT 16ull
```

Alignment requirement for OptixBuildInputInstanceArray::instances.

5.12.2.8 OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
```

Alignment requirement for opacity micromap array buffers.

5.12.2.9 OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL

```
#define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
```

Maximum subdivision level for opacity micromaps.

5.12.2.10 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE

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

5.12.2.11 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.

5.12.2.12 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE

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

5.12.2.13 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPAREN

```
#define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_
TRANSPARENT (-3)
```

5.12.2.14 OPTIX_OPACITY_MICROMAP_STATE_OPAQUE

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

5.12.2.15 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.

5.12.2.16 OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE

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

5.12.2.17 OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT

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

5.12.2.18 OPTIX_SBT_RECORD_ALIGNMENT

```
#define OPTIX_SBT_RECORD_ALIGNMENT 16ull
```

Alignment requirement for device pointers in OptixShaderBindingTable.

5.12.2.19 OPTIX_SBT_RECORD_HEADER_SIZE

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

Size of the SBT record headers.

5.12.2.20 OPTIX_TRANSFORM_BYTE_ALIGNMENT

```
#define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
```

Alignment requirement for OptixStaticTransform, OptixMatrixMotionTransform, OptixSRTMotionTransform.

5.12.3 Typedef Documentation

5.12.3.1 CUdeviceptr

```
typedef unsigned long long CUdeviceptr
```

CUDA device pointer.

5.12.3.2 OptixAabb

```
typedef struct OptixAabb OptixAabb
```

AABB inputs.

5.12.3.3 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( )

5.12.3.4 OptixAccelBuildOptions

`typedef struct OptixAccelBuildOptions OptixAccelBuildOptions`

Build options for acceleration structures.

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( )

5.12.3.5 OptixAccelEmitDesc

`typedef struct OptixAccelEmitDesc OptixAccelEmitDesc`

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

See also optixAccelBuild( )

5.12.3.6 OptixAccelPropertyType

`typedef enum OptixAccelPropertyType OptixAccelPropertyType`

Properties which can be emitted during acceleration structure build.

See also OptixAccelEmitDesc::type.

5.12.3.7 OptixBuildFlags

`typedef enum OptixBuildFlags OptixBuildFlags`

Builder Options.

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

5.12.3.8 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( )

5.12.3.9 OptixBuildInputCurveArray

`typedef struct OptixBuildInputCurveArray OptixBuildInputCurveArray`

Curve inputs.

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

OptiX describes the curve array as a list of curve segments. The primitive id is the segment number. It is the user's responsibility to maintain a mapping between curves and curve segments. Each index

buffer entry i = indexBuffer[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

5.12.3.10 OptixBuildInputCustomPrimitiveArray

```
typedef struct OptixBuildInputCustomPrimitiveArray
OptixBuildInputCustomPrimitiveArray
```

Custom primitive inputs.

See also OptixBuildInput::customPrimitiveArray

5.12.3.11 OptixBuildInputInstanceArray

```
typedef struct OptixBuildInputInstanceArray OptixBuildInputInstanceArray
```

Instance and instance pointer inputs.

See also OptixBuildInput::instanceArray

5.12.3.12 OptixBuildInputOpacityMicromap

```
typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap
```

5.12.3.13 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

5.12.3.14 OptixBuildInputTriangleArray

```
typedef struct OptixBuildInputTriangleArray OptixBuildInputTriangleArray
```

Triangle inputs.

See also OptixBuildInput::triangleArray

5.12.3.15 OptixBuildInputType

```
typedef enum OptixBuildInputType OptixBuildInputType
```

Enum to distinguish the different build input types.

See also OptixBuildInput::type

5.12.3.16 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

5.12.3.17 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( )

5.12.3.18 OptixCompileDebugLevel

```
typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
```

Debug levels.

See also OptixModuleCompileOptions::debugLevel

5.12.3.19 OptixCompileOptimizationLevel

```
typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
```

Optimization levels.

See also OptixModuleCompileOptions::optLevel

5.12.3.20 OptixCurveEndcapFlags

```
typedef enum OptixCurveEndcapFlags OptixCurveEndcapFlags
```

Curve end cap types, for non-linear curves.

5.12.3.21 OptixDenoiser

```
typedef struct OptixDenoiser_t* OptixDenoiser
```

Opaque type representing a denoiser instance.

5.12.3.22 OptixDenoiserAlphaMode

```
typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
```

Various parameters used by the denoiser.

See also optixDenoiserInvoke( )

optixDenoiserComputeIntensity( )

optixDenoiserComputeAverageColor( )

5.12.3.23 OptixDenoiserGuideLayer

```
typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
```

Guide layer for the denoiser.

See also optixDenoiserInvoke( )

5.12.3.24 OptixDenoiserLayer

```
typedef struct OptixDenoiserLayer OptixDenoiserLayer
```

Input/Output layers for the denoiser.

See also optixDenoiserInvoke( )

5.12.3.25 OptixDenoiserModelKind

```
typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
```

Model kind used by the denoiser.

See also optixDenoiserCreate

5.12.3.26 OptixDenoiserOptions

```
typedef struct OptixDenoiserOptions OptixDenoiserOptions
```

Options used by the denoiser.

See also optixDenoiserCreate( )

5.12.3.27 OptixDenoiserParams

```
typedef struct OptixDenoiserParams OptixDenoiserParams
```

5.12.3.28 OptixDenoiserSizes

```
typedef struct OptixDenoiserSizes OptixDenoiserSizes
```

Various sizes related to the denoiser.

See also optixDenoiserComputeMemoryResources( )

5.12.3.29 OptixDeviceContext

```
typedef struct OptixDeviceContext_t* OptixDeviceContext
```

Opaque type representing a device context.

5.12.3.30 OptixDeviceContextOptions

```
typedef struct OptixDeviceContextOptions OptixDeviceContextOptions
```

Parameters used for optixDeviceContextCreate( )

See also optixDeviceContextCreate( )

5.12.3.31 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( )

5.12.3.32 OptixDeviceProperty

`typedef enum OptixDeviceProperty OptixDeviceProperty`

Parameters used for optixDeviceContextGetProperty( )

See also optixDeviceContextGetProperty( )

5.12.3.33 OptixExceptionCodes

`typedef enum OptixExceptionCodes OptixExceptionCodes`

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

5.12.3.34 OptixExceptionFlags

`typedef enum OptixExceptionFlags OptixExceptionFlags`

Exception flags.

See also OptixPipelineCompileOptions::exceptionFlags, OptixExceptionCodes

5.12.3.35 OptixGeometryFlags

`typedef enum OptixGeometryFlags OptixGeometryFlags`

Flags used by OptixBuildInputTriangleArray::flags and #OptixBuildInput::flag and OptixBuildInputCustomPrimitiveArray::flags.

5.12.3.36 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( )

5.12.3.37 OptixImage2D

`typedef struct OptixImage2D OptixImage2D`

Image descriptor used by the denoiser.

See also optixDenoiserInvoke( ), optixDenoiserComputeIntensity( )

5.12.3.38 OptixIndicesFormat

`typedef enum OptixIndicesFormat OptixIndicesFormat`

Format of indices used int OptixBuildInputTriangleArray::indexFormat.

5.12.3.39 OptixInstance

`typedef struct OptixInstance OptixInstance`

Instances.

See also OptixBuildInputInstanceArray::instances

5.12.3.40 OptixInstanceFlags

`typedef enum OptixInstanceFlags OptixInstanceFlags`

Flags set on the OptixInstance::flags.

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

5.12.3.41 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	*level*	The log level indicates the severity of the message. See below for possible values.
in	*tag*	A terse message category description (e.g., 'SCENE STAT').
in	*message*	Null terminated log message (without newline at the end).
in	*cbdata*	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

5.12.3.42 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( )

5.12.3.43 OptixMicromapBuffers

```
typedef struct OptixMicromapBuffers OptixMicromapBuffers
```

Buffer inputs for opacity micromap array builds.

5.12.3.44 OptixMicromapBufferSizes

```
typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
```

Conservative memory requirements for building a opacity micromap array.

5.12.3.45 OptixModule

```
typedef struct OptixModule_t* OptixModule
```

Opaque type representing a module.

5.12.3.46 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 consants 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 optixModuleCreateFromPTX 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 optixModuleCreateFromPTX.

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

5.12.3.47 OptixModuleCompileOptions

```
typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
```

Compilation options for module.

See also optixModuleCreateFromPTX( )

5.12.3.48 OptixModuleCompileState

```
typedef enum OptixModuleCompileState OptixModuleCompileState
```

Module compilation state.

See also optixModuleGetCompilationState( ), optixModuleCreateFromPTXWithTasks( )

5.12.3.49 OptixMotionFlags

```
typedef enum OptixMotionFlags OptixMotionFlags
```

Enum to specify motion flags.

See also OptixMotionOptions::flags.

5.12.3.50 OptixMotionOptions

```
typedef struct OptixMotionOptions OptixMotionOptions
```

Motion options.

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

5.12.3.51 OptixOpacityMicromapArrayBuildInput

```
typedef struct OptixOpacityMicromapArrayBuildInput
OptixOpacityMicromapArrayBuildInput
```

Inputs to opacity micromap array construction.

5.12.3.52 OptixOpacityMicromapArrayIndexingMode

```
typedef enum OptixOpacityMicromapArrayIndexingMode
OptixOpacityMicromapArrayIndexingMode
```

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

5.12.3.53 OptixOpacityMicromapDesc

```
typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc
```

Opacity micromap descriptor.

5.12.3.54 OptixOpacityMicromapFlags

```
typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags
```

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

5.12.3.55 OptixOpacityMicromapFormat

```
typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat
```

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

5.12.3.56 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.

5.12.3.57 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.

5.12.3.58 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.

5.12.3.59 OptixPayloadType

```
typedef struct OptixPayloadType OptixPayloadType
```

Specifies a single payload type.

5.12.3.60 OptixPayloadTypeID

```
typedef enum OptixPayloadTypeID OptixPayloadTypeID
```

Payload type identifiers.

5.12.3.61 OptixPipeline

```
typedef struct OptixPipeline_t* OptixPipeline
```

Opaque type representing a pipeline.

5.12.3.62 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 optixModuleCreateFromPTX( ), optixPipelineCreate( )

5.12.3.63 OptixPipelineLinkOptions

```
typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
```

Link options for a pipeline.

See also optixPipelineCreate( )

5.12.3.64 OptixPixelFormat

```
typedef enum OptixPixelFormat OptixPixelFormat
```

Pixel formats used by the denoiser.

See also OptixImage2D::format

5.12.3.65 OptixPrimitiveType

```
typedef enum OptixPrimitiveType OptixPrimitiveType
```

Builtin primitive types.

5.12.3.66 OptixPrimitiveTypeFlags

```
typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags
```

Builtin flags may be bitwise combined.

See also OptixPipelineCompileOptions::usesPrimitiveTypeFlags

5.12.3.67 OptixProgramGroup

```
typedef struct OptixProgramGroup_t* OptixProgramGroup
```

Opaque type representing a program group.

5.12.3.68 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

5.12.3.69 OptixProgramGroupDesc

```
typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
```

Descriptor for program groups.

5.12.3.70 OptixProgramGroupFlags

```
typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
```

Flags for program groups.

5.12.3.71 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

5.12.3.72 OptixProgramGroupKind

```
typedef enum OptixProgramGroupKind OptixProgramGroupKind
```

Distinguishes different kinds of program groups.

5.12.3.73 OptixProgramGroupOptions

```
typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
```

Program group options.

See also optixProgramGroupCreate( )

5.12.3.74 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

5.12.3.75 OptixQueryFunctionTable_t

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

Type of the function `optixQueryFunctionTable( )`

5.12.3.76 OptixQueryFunctionTableOptions

```
typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
```

Options that can be passed to `optixQueryFunctionTable( )`

5.12.3.77 OptixRayFlags

```
typedef enum OptixRayFlags OptixRayFlags
```

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

See also optixTrace( )

5.12.3.78 OptixRelocateInput

```
typedef struct OptixRelocateInput OptixRelocateInput
```

Relocation inputs.

See also optixAccelRelocate( )

5.12.3.79 OptixRelocateInputInstanceArray

```
typedef struct OptixRelocateInputInstanceArray
OptixRelocateInputInstanceArray
```

Instance and instance pointer inputs.

See also OptixRelocateInput::instanceArray

5.12.3.80 OptixRelocateInputOpacityMicromap

```
typedef struct OptixRelocateInputOpacityMicromap
OptixRelocateInputOpacityMicromap
```

5.12.3.81 OptixRelocateInputTriangleArray

```
typedef struct OptixRelocateInputTriangleArray
OptixRelocateInputTriangleArray
```

Triangle inputs.

See also OptixRelocateInput::triangleArray

5.12.3.82 OptixRelocationInfo

```
typedef struct OptixRelocationInfo OptixRelocationInfo
```

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

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

5.12.3.83 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( )

5.12.3.84 OptixShaderBindingTable

```
typedef struct OptixShaderBindingTable OptixShaderBindingTable
```

Describes the shader binding table (SBT)

See also optixLaunch( )

5.12.3.85 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( )

5.12.3.86 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( )

5.12.3.87 OptixStackSizes

`typedef struct OptixStackSizes OptixStackSizes`

Describes the stack size requirements of a program group.

See also optixProgramGroupGetStackSize( )

5.12.3.88 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( )

5.12.3.89 OptixTask

```
typedef struct OptixTask_t* OptixTask
```

Opaque type representing a work task.

5.12.3.90 OptixTransformFormat

```
typedef enum OptixTransformFormat OptixTransformFormat
```

Format of transform used in OptixBuildInputTriangleArray::transformFormat.

5.12.3.91 OptixTransformType

```
typedef enum OptixTransformType OptixTransformType
```

Transform.

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

5.12.3.92 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.

5.12.3.93 OptixTraversableHandle

```
typedef unsigned long long OptixTraversableHandle
```

Traversable handle.

5.12.3.94 OptixTraversableType

```
typedef enum OptixTraversableType OptixTraversableType
```

Traversable Handles.

See also optixConvertPointerToTraversableHandle( )

5.12.3.95 OptixVertexFormat

```
typedef enum OptixVertexFormat OptixVertexFormat
```

Format of vertices used in OptixBuildInputTriangleArray::vertexFormat.

5.12.3.96 OptixVisibilityMask

```
typedef unsigned int OptixVisibilityMask
```

Visibility mask.

5.12.4 Enumeration Type Documentation

5.12.4.1 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	OptixAabb * numMotionSteps.

5.12.4.2 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 optixAccelBuild.
OPTIX_BUILD_FLAG_ALLOW_COMPACTION	
OPTIX_BUILD_FLAG_PREFER_FAST_TRACE	
OPTIX_BUILD_FLAG_PREFER_FAST_BUILD	
OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS	Allow random access to build input vertices See optixGetTriangleVertexData optixGetLinearCurveVertexData optixGetQuadraticBSplineVertexData optixGetCubicBSplineVertexData optixGetCatmullRomVertexData optixGetSphereData.
OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS	Allow random access to instances See optixGetInstanceTraversableFromIAS.
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.

Enumerator

OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS	If enabled, any instances referencing this GAS are allowed to disable the opacity micromap test through the DISABLE_OPACITY_MICROMAPS flag instance flag. Note that the GAS will not be optimized for the attached opacity micromap Arrays if this flag is set, which may result in reduced traversal performance.

5.12.4.3 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 OptixBuildInputTriangleArray
OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES	Custom primitive inputs. See also OptixBuildInputCustomPrimitiveArray
OPTIX_BUILD_INPUT_TYPE_INSTANCES	Instance inputs. See also OptixBuildInputInstanceArray
OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS	Instance pointer inputs. See also OptixBuildInputInstanceArray
OPTIX_BUILD_INPUT_TYPE_CURVES	Curve inputs. See also OptixBuildInputCurveArray
OPTIX_BUILD_INPUT_TYPE_SPHERES	Sphere inputs. See also OptixBuildInputSphereArray

5.12.4.4 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.

5.12.4.5 OptixCompileDebugLevel

`enum OptixCompileDebugLevel`

Debug levels.

See also OptixModuleCompileOptions::debugLevel

Enumerator

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

5.12.4.6 OptixCompileOptimizationLevel

`enum OptixCompileOptimizationLevel`

Optimization levels.

See also OptixModuleCompileOptions::optLevel

Enumerator

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

5.12.4.7 OptixCurveEndcapFlags

`enum OptixCurveEndcapFlags`

Curve end cap types, for non-linear curves.

Enumerator

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

5.12.4.8 OptixDenoiserAlphaMode

`enum OptixDenoiserAlphaMode`

Various parameters used by the denoiser.

See also optixDenoiserInvoke( )

optixDenoiserComputeIntensity( )

optixDenoiserComputeAverageColor( )

Enumerator

OPTIX_DENOISER_ALPHA_MODE_COPY	Copy alpha (if present) from input layer, no denoising.
OPTIX_DENOISER_ALPHA_MODE_ALPHA_AS_AOV	Denoise alpha separately. With AOV model kinds, treat alpha like an AOV.
OPTIX_DENOISER_ALPHA_MODE_FULL_DENOISE_PASS	With AOV model kinds, full denoise pass with alpha. This is slower than OPTIX_DENOISER_ALPHA_MODE_ALPHA_AS_AOV.

5.12.4.9 OptixDenoiserModelKind

`enum OptixDenoiserModelKind`

Model kind used by the denoiser.

See also optixDenoiserCreate

Enumerator

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

5.12.4.10 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

5.12.4.11 OptixDeviceProperty

`enum OptixDeviceProperty`

Parameters used for optixDeviceContextGetProperty( )

See also optixDeviceContextGetProperty( )

Enumerator

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

5.12.4.12 OptixExceptionCodes

`enum OptixExceptionCodes`

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

Enumerator

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

Enumerator

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

5.12.4.13 OptixExceptionFlags

`enum OptixExceptionFlags`

Exception flags.

See also OptixPipelineCompileOptions::exceptionFlags, OptixExceptionCodes

Enumerator

OPTIX_EXCEPTION_FLAG_NONE	No exception are enabled.

Enumerator

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

5.12.4.14 OptixGeometryFlags

`enum OptixGeometryFlags`

Flags used by OptixBuildInputTriangleArray::flags and #OptixBuildInput::flag 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.

5.12.4.15 OptixHitKind

`enum OptixHitKind`

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

See also optixGetHitKind( )

Enumerator

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

5.12.4.16 OptixIndicesFormat

`enum OptixIndicesFormat`

Format of indices used int OptixBuildInputTriangleArray::indexFormat.

Enumerator

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

5.12.4.17 OptixInstanceFlags

`enum OptixInstanceFlags`

Flags set on the OptixInstance::flags.

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

Enumerator

OPTIX_INSTANCE_FLAG_NONE	No special flag set.
OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING	Prevent triangles from getting culled due to their orientation. Effectively ignores ray flags OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES and OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES.
OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING	Flip triangle orientation. This affects front/backface culling as well as the reported face in case of a hit.
OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT	Disable anyhit programs for all geometries of the instance. Can be overridden by OPTIX_RAY_FLAG_ENFORCE_ANYHIT. This flag is mutually exclusive with OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT.
OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT	Enables anyhit programs for all geometries of the instance. Overrides OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT Can be overridden by OPTIX_RAY_FLAG_DISABLE_ANYHIT. This flag is mutually exclusive with OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT.
OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE	Force 4-state opacity micromaps to behave as 2-state opacity micromaps during traversal.
OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS	Don't perform opacity micromap query for this instance. GAS must be built with ALLOW_DISABLE_OPACITY_MICROMAPS for this to be valid. This flag overrides FORCE_OPACTIY_MIXROMAP_2_STATE instance and ray flags.

5.12.4.18 OptixModuleCompileState

`enum OptixModuleCompileState`

Module compilation state.

See also optixModuleGetCompilationState( ), optixModuleCreateFromPTXWithTasks( )

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.

5.12.4.19 OptixMotionFlags

`enum OptixMotionFlags`

Enum to specify motion flags.

See also OptixMotionOptions::flags.

Enumerator

OPTIX_MOTION_FLAG_NONE
OPTIX_MOTION_FLAG_START_VANISH
OPTIX_MOTION_FLAG_END_VANISH

5.12.4.20 OptixOpacityMicromapArrayIndexingMode

`enum OptixOpacityMicromapArrayIndexingMode`

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

Enumerator

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

Enumerator

OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED	OptixBuildInputVisibleMap::indexBuffer provides a per triangle array of predefined indices and/or indices into OptixBuildInputVisibleMap ::opacityMicromapArray. See OptixBuildInputOpacityMicromap::indexBuffer for more details.

5.12.4.21 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	
OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD	

5.12.4.22 OptixOpacityMicromapFormat

`enum OptixOpacityMicromapFormat`

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

Enumerator

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

5.12.4.23 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

5.12.4.24 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	

5.12.4.25 OptixPixelFormat

`enum OptixPixelFormat`

Pixel formats used by the denoiser.

See also OptixImage2D::format

Enumerator

OPTIX_PIXEL_FORMAT_HALF2	two halfs, XY
OPTIX_PIXEL_FORMAT_HALF3	three halfs, RGB
OPTIX_PIXEL_FORMAT_HALF4	four halfs, RGBA
OPTIX_PIXEL_FORMAT_FLOAT2	two floats, XY
OPTIX_PIXEL_FORMAT_FLOAT3	three floats, RGB
OPTIX_PIXEL_FORMAT_FLOAT4	four floats, RGBA
OPTIX_PIXEL_FORMAT_UCHAR3	three unsigned chars, RGB
OPTIX_PIXEL_FORMAT_UCHAR4	four unsigned chars, RGBA
OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER	internal format

5.12.4.26 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_SPHERE	
OPTIX_PRIMITIVE_TYPE_TRIANGLE	Triangle.

5.12.4.27 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.

Enumerator

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_SPHERE	
OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE	Triangle.

5.12.4.28 OptixProgramGroupFlags

`enum OptixProgramGroupFlags`

Flags for program groups.

Enumerator

OPTIX_PROGRAM_GROUP_FLAGS_NONE	Currently there are no flags.

5.12.4.29 OptixProgramGroupKind

`enum OptixProgramGroupKind`

Distinguishes different kinds of program groups.

Enumerator

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

5.12.4.30 OptixQueryFunctionTableOptions

`enum OptixQueryFunctionTableOptions`

Options that can be passed to `optixQueryFunctionTable( )`

Enumerator

OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY	Placeholder (there are no options yet)

5.12.4.31 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.
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.

Enumerator

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 opactiy micromaps during traversal.

5.12.4.32 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

Enumerator

OPTIX_ERROR_CUDA_NOT_INITIALIZED
OPTIX_ERROR_VALIDATION_FAILURE
OPTIX_ERROR_INVALID_PTX
OPTIX_ERROR_INVALID_LAUNCH_PARAMETER
OPTIX_ERROR_INVALID_PAYLOAD_ACCESS
OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS
OPTIX_ERROR_INVALID_FUNCTION_USE
OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS
OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY
OPTIX_ERROR_PIPELINE_LINK_ERROR
OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE
OPTIX_ERROR_INTERNAL_COMPILER_ERROR
OPTIX_ERROR_DENOISER_MODEL_NOT_SET
OPTIX_ERROR_DENOISER_NOT_INITIALIZED
OPTIX_ERROR_NOT_COMPATIBLE
OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH
OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED
OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID
OPTIX_ERROR_NOT_SUPPORTED
OPTIX_ERROR_UNSUPPORTED_ABI_VERSION
OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH
OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS
OPTIX_ERROR_LIBRARY_NOT_FOUND
OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND
OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE
OPTIX_ERROR_DEVICE_OUT_OF_MEMORY
OPTIX_ERROR_CUDA_ERROR
OPTIX_ERROR_INTERNAL_ERROR
OPTIX_ERROR_UNKNOWN

5.12.4.33 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

5.12.4.34 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 OptixStaticTransform
OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM	See also OptixMatrixMotionTransform
OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM	See also OptixSRTMotionTransform
OPTIX_TRANSFORM_TYPE_INSTANCE	See also OptixInstance

5.12.4.35 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.

5.12.4.36 OptixTraversableType

`enum OptixTraversableType`

Traversable Handles.

See also optixConvertPointerToTraversableHandle( )

Enumerator

OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM	Static transforms. See also OptixStaticTransform
OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM	Matrix motion transform. See also OptixMatrixMotionTransform
OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM	SRT motion transform. See also OptixSRTMotionTransform

5.12.4.37 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 halfs.
OPTIX_VERTEX_FORMAT_HALF2	Vertices are represented by two halfs.
OPTIX_VERTEX_FORMAT_SNORM16_3	
OPTIX_VERTEX_FORMAT_SNORM16_2	

5.12.5 Variable Documentation

5.12.5.1

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

5.12.5.2

`union { ... } OptixRelocateInput::@3`

5.12.5.3

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

5.12.5.4 a

`float OptixSRTData::a`

5.12.5.5 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.

5.12.5.6 albedo

`OptixImage2D OptixDenoiserGuideLayer::albedo`

5.12.5.7 allowOpacityMicromaps

`int OptixPipelineCompileOptions::allowOpacityMicromaps`

Boolean value indicating whether opacity micromaps could be used.

5.12.5.8 annotation

`const char* OptixModuleCompileBoundValueEntry::annotation`

5.12.5.9 b

`float OptixSRTData::b`

5.12.5.10 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.

5.12.5.11 boundValuePtr

`const void* OptixModuleCompileBoundValueEntry::boundValuePtr`

5.12.5.12 boundValues

`const OptixModuleCompileBoundValueEntry* OptixModuleCompileOptions ::boundValues`

Ingored if numBoundValues is set to 0.

5.12.5.13 buildFlags [1/2]

`unsigned int OptixAccelBuildOptions::buildFlags`

Combinations of OptixBuildFlags.

5.12.5.14 buildFlags [2/2]

`unsigned int OptixBuiltinISOptions::buildFlags`

Build flags, see OptixBuildFlags.

5.12.5.15 builtinISModuleType

`OptixPrimitiveType OptixBuiltinISOptions::builtinISModuleType`

5.12.5.16 byteOffset

`unsigned int OptixOpacityMicromapDesc::byteOffset`

Byte offset to opacity micromap in data input buffer of opacity micromap array build.

5.12.5.17 c

`float OptixSRTData::c`

5.12.5.18 callables [1/2]

`OptixProgramGroupCallables OptixProgramGroupDesc::callables`

See also OPTIX_PROGRAM_GROUP_KIND_CALLABLES

5.12.5.19 [2/2]

`OptixProgramGroupCallables { ... } ::callables`

See also OPTIX_PROGRAM_GROUP_KIND_CALLABLES

5.12.5.20 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.

5.12.5.21 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.

5.12.5.22 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.

5.12.5.23 child [1/3]

`OptixTraversableHandle OptixStaticTransform::child`

The traversable transformed by this transformation.

5.12.5.24 child [2/3]

`OptixTraversableHandle OptixMatrixMotionTransform::child`

The traversable that is transformed by this transformation.

5.12.5.25 child [3/3]

`OptixTraversableHandle OptixSRTMotionTransform::child`

The traversable transformed by this transformation.

5.12.5.26 computeAverageColorSizeInBytes

`size_t OptixDenoiserSizes::computeAverageColorSizeInBytes`

Size of scratch memory passed to optixDenoiserComputeAverageColor. The size is independent of the tile/image resolution.

5.12.5.27 computeIntensitySizeInBytes

`size_t OptixDenoiserSizes::computeIntensitySizeInBytes`

Size of scratch memory passed to optixDenoiserComputeIntensity. The size is independent of the tile/image resolution.

5.12.5.28 count [1/2]

`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.

5.12.5.29 count [2/2]

`unsigned int OptixOpacityMicromapHistogramEntry::count`

Number of opacity micromaps with the format and subdivision level that are input to the opacity micromap array build.

5.12.5.30 cssAH

`unsigned int OptixStackSizes::cssAH`

Continuation stack size of AH programs in bytes.

5.12.5.31 cssCC

`unsigned int OptixStackSizes::cssCC`

Continuation stack size of CC programs in bytes.

5.12.5.32 cssCH

`unsigned int OptixStackSizes::cssCH`

Continuation stack size of CH programs in bytes.

5.12.5.33 cssIS

`unsigned int OptixStackSizes::cssIS`

Continuation stack size of IS programs in bytes.

5.12.5.34 cssMS

`unsigned int OptixStackSizes::cssMS`

Continuation stack size of MS programs in bytes.

5.12.5.35 cssRG

`unsigned int OptixStackSizes::cssRG`

Continuation stack size of RG programs in bytes.

5.12.5.36 [1/2]

`OptixBuildInputCurveArray { ... } ::curveArray`

Curve inputs.

5.12.5.37 curveArray [2/2]

`OptixBuildInputCurveArray OptixBuildInput::curveArray`

Curve inputs.

5.12.5.38 curveEndcapFlags

`unsigned int OptixBuiltinISOptions::curveEndcapFlags`

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

5.12.5.39 curveType

`OptixPrimitiveType OptixBuildInputCurveArray::curveType`

Curve degree and basis.

See also OptixPrimitiveType

5.12.5.40 [1/2]

`OptixBuildInputCustomPrimitiveArray { ... } ::customPrimitiveArray`

Custom primitive inputs.

5.12.5.41 customPrimitiveArray [2/2]

`OptixBuildInputCustomPrimitiveArray OptixBuildInput::customPrimitiveArray`

Custom primitive inputs.

5.12.5.42 data

`CUdeviceptr OptixImage2D::data`

Pointer to the actual pixel data.

5.12.5.43 debugLevel [1/2]

`OptixCompileDebugLevel OptixModuleCompileOptions::debugLevel`

Generate debug information.

5.12.5.44 debugLevel [2/2]

`OptixCompileDebugLevel OptixPipelineLinkOptions::debugLevel`

Generate debug information.

5.12.5.45 denoiseAlpha

`OptixDenoiserAlphaMode OptixDenoiserParams::denoiseAlpha`

alpha denoise mode

5.12.5.46 dssDC

`unsigned int OptixStackSizes::dssDC`

Direct stack size of DC programs in bytes.

5.12.5.47 endcapFlags

`unsigned int OptixBuildInputCurveArray::endcapFlags`

End cap flags, see OptixCurveEndcapFlags.

5.12.5.48 entryFunctionName

`const char* OptixProgramGroupSingleModule::entryFunctionName`

Entry function name of the single program.

5.12.5.49 entryFunctionNameAH

`const char* OptixProgramGroupHitgroup::entryFunctionNameAH`

Entry function name of the any hit (AH) program.

5.12.5.50 entryFunctionNameCC

`const char* OptixProgramGroupCallables::entryFunctionNameCC`

Entry function name of the continuation callable (CC) program.

5.12.5.51 entryFunctionNameCH

`const char* OptixProgramGroupHitgroup::entryFunctionNameCH`

Entry function name of the closest hit (CH) program.

5.12.5.52 entryFunctionNameDC

`const char* OptixProgramGroupCallables::entryFunctionNameDC`

Entry function name of the direct callable (DC) program.

5.12.5.53 entryFunctionNameIS

`const char* OptixProgramGroupHitgroup::entryFunctionNameIS`

Entry function name of the intersection (IS) program.

5.12.5.54 exception [1/2]

`OptixProgramGroupSingleModule OptixProgramGroupDesc::exception`

See also OPTIX_PROGRAM_GROUP_KIND_EXCEPTION

5.12.5.55 [2/2]

`OptixProgramGroupSingleModule { ... } ::exception`

See also OPTIX_PROGRAM_GROUP_KIND_EXCEPTION

5.12.5.56 exceptionFlags

`unsigned int OptixPipelineCompileOptions::exceptionFlags`

A bitmask of OptixExceptionFlags indicating which exceptions are enabled.

5.12.5.57 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.

5.12.5.58 flag

`unsigned int OptixBuildInputCurveArray::flag`

Combination of OptixGeometryFlags describing the primitive behavior.

5.12.5.59 flags [1/7]

`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.

5.12.5.60 flags [2/7]

`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.

5.12.5.61 flags [3/7]

`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.

5.12.5.62 flags [4/7]

`unsigned int OptixInstance::flags`

Any combination of OptixInstanceFlags is allowed.

5.12.5.63 flags [5/7]

`OptixOpacityMicromapFlags OptixOpacityMicromapArrayBuildInput::flags`

Applies to all opacity micromaps in array.

5.12.5.64 flags [6/7]

`unsigned short OptixMotionOptions::flags`

Combinations of OptixMotionFlags.

5.12.5.65 flags [7/7]

`unsigned int OptixProgramGroupDesc::flags`

See OptixProgramGroupFlags.

5.12.5.66 flow

`OptixImage2D OptixDenoiserGuideLayer::flow`

5.12.5.67 format [1/4]

`OptixOpacityMicromapFormat OptixOpacityMicromapUsageCount::format`

opacity micromap format.

5.12.5.68 format [2/4]

`unsigned short OptixOpacityMicromapDesc::format`

OptixOpacityMicromapFormat.

5.12.5.69 format [3/4]

`OptixOpacityMicromapFormat OptixOpacityMicromapHistogramEntry::format`

opacity micromap format.

5.12.5.70 format [4/4]

`OptixPixelFormat OptixImage2D::format`

Pixel format.

5.12.5.71 guideAlbedo

`unsigned int OptixDenoiserOptions::guideAlbedo`

5.12.5.72 guideNormal

`unsigned int OptixDenoiserOptions::guideNormal`

5.12.5.73 hdrAverageColor

`CUdeviceptr OptixDenoiserParams::hdrAverageColor`

this parameter is used when the OPTIX_DENOISER_MODEL_KIND_AOV model kind is set. average log color of input image, separate for RGB channels (default null pointer). points to three floats. with the default (null pointer) denoised results will not be optimal.

5.12.5.74 hdrIntensity

`CUdeviceptr OptixDenoiserParams::hdrIntensity`

average log intensity of input image (default null pointer). points to a single float. with the default (null pointer) denoised results will not be optimal for very dark or bright input images.

5.12.5.75 height

`unsigned int OptixImage2D::height`

Height of the image (in pixels)

5.12.5.76 hitgroup [1/2]

`OptixProgramGroupHitgroup OptixProgramGroupDesc::hitgroup`

See also OPTIX_PROGRAM_GROUP_KIND_HITGROUP

5.12.5.77 [2/2]

`OptixProgramGroupHitgroup { ... } ::hitgroup`

See also OPTIX_PROGRAM_GROUP_KIND_HITGROUP

5.12.5.78 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.

5.12.5.79 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.

5.12.5.80 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.

5.12.5.81 indexBuffer [1/3]

`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.

5.12.5.82 indexBuffer [2/3]

`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.

5.12.5.83 indexBuffer [3/3]

`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.

5.12.5.84 indexFormat

`OptixIndicesFormat OptixBuildInputTriangleArray::indexFormat`

See also OptixIndicesFormat

5.12.5.85 indexingMode

`OptixOpacityMicromapArrayIndexingMode OptixBuildInputOpacityMicromap ::indexingMode`

Indexing mode of triangle to opacity micromap array mapping.

5.12.5.86 indexOffset

`unsigned int OptixBuildInputOpacityMicromap::indexOffset`

Constant offset to non-negative opacity micromap indices.

5.12.5.87 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.

5.12.5.88 indexStrideInBytes [1/3]

`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.

5.12.5.89 indexStrideInBytes [2/3]

`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.

5.12.5.90 indexStrideInBytes [3/3]

`unsigned int OptixBuildInputCurveArray::indexStrideInBytes`

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

5.12.5.91 info

`unsigned long long OptixRelocationInfo::info[4]`

Opaque data, used internally, should not be modified.

5.12.5.92 input

```
OptixImage2D OptixDenoiserLayer::input
```

5.12.5.93 inputBuffer

```
CUdeviceptr OptixOpacityMicromapArrayBuildInput::inputBuffer
```

128B aligned base pointer for raw opacity micromap input data.

5.12.5.94 instanceArray [1/4]

```
OptixBuildInputInstanceArray OptixBuildInput::instanceArray
```

Instance and instance pointer inputs.

5.12.5.95 [2/4]

```
OptixBuildInputInstanceArray { ... } ::instanceArray
```

Instance and instance pointer inputs.

5.12.5.96 instanceArray [3/4]

```
OptixRelocateInputInstanceArray OptixRelocateInput::instanceArray
```

Instance and instance pointer inputs.

5.12.5.97 [4/4]

```
OptixRelocateInputInstanceArray { ... } ::instanceArray
```

Instance and instance pointer inputs.

5.12.5.98 instanceId

```
unsigned int OptixInstance::instanceId
```

Application supplied ID. The maximal ID can be queried using OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID.

5.12.5.99 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.

5.12.5.100 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)

5.12.5.101 internalGuideLayerPixelSizeInBytes

`size_t OptixDenoiserSizes::internalGuideLayerPixelSizeInBytes`

Number of bytes for each pixel in internal guide layers.

5.12.5.102 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.

5.12.5.103 kind

`OptixProgramGroupKind OptixProgramGroupDesc::kind`

The kind of program group.

5.12.5.104 logCallbackData

`void* OptixDeviceContextOptions::logCallbackData`

Pointer stored and passed to logCallbackFunction when a message is generated.

5.12.5.105 logCallbackFunction

`OptixLogCallback OptixDeviceContextOptions::logCallbackFunction`

Function pointer used when OptiX wishes to generate messages.

5.12.5.106 logCallbackLevel

`int OptixDeviceContextOptions::logCallbackLevel`

Maximum callback level to generate message for (see OptixLogCallback)

5.12.5.107 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.

5.12.5.108 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.

5.12.5.109 maxX

`float OptixAabb::maxX`

Upper extent in X direction.

5.12.5.110 maxY

`float OptixAabb::maxY`

Upper extent in Y direction.

5.12.5.111 maxZ

`float OptixAabb::maxZ`

Upper extent in Z direction.

5.12.5.112 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.

5.12.5.113 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.

5.12.5.114 minX

`float OptixAabb::minX`

Lower extent in X direction.

5.12.5.115 minY

`float OptixAabb::minY`

Lower extent in Y direction.

5.12.5.116 minZ

`float OptixAabb::minZ`

Lower extent in Z direction.

5.12.5.117 [1/2]

`OptixProgramGroupSingleModule { ... } ::miss`

See also OPTIX_PROGRAM_GROUP_KIND_MISS

5.12.5.118 miss [2/2]

`OptixProgramGroupSingleModule OptixProgramGroupDesc::miss`

See also OPTIX_PROGRAM_GROUP_KIND_MISS

5.12.5.119 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.

5.12.5.120 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.

5.12.5.121 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.

5.12.5.122 module

`OptixModule OptixProgramGroupSingleModule::module`

Module holding single program.

5.12.5.123 moduleAH

`OptixModule OptixProgramGroupHitgroup::moduleAH`

Module holding the any hit (AH) program.

5.12.5.124 moduleCC

`OptixModule OptixProgramGroupCallables::moduleCC`

Module holding the continuation callable (CC) program.

5.12.5.125 moduleCH

`OptixModule OptixProgramGroupHitgroup::moduleCH`

Module holding the closest hit (CH) program.

5.12.5.126 moduleDC

`OptixModule OptixProgramGroupCallables::moduleDC`

Module holding the direct callable (DC) program.

5.12.5.127 moduleIS

`OptixModule OptixProgramGroupHitgroup::moduleIS`

Module holding the intersection (Is) program.

5.12.5.128 motionOptions [1/3]

`OptixMotionOptions OptixAccelBuildOptions::motionOptions`

Options for motion.

5.12.5.129 motionOptions [2/3]

`OptixMotionOptions OptixMatrixMotionTransform::motionOptions`

The motion options for this transformation. Must have at least two motion keys.

5.12.5.130 motionOptions [3/3]

`OptixMotionOptions OptixSRTMotionTransform::motionOptions`

The motion options for this transformation Must have at least two motion keys.

5.12.5.131 normal

`OptixImage2D OptixDenoiserGuideLayer::normal`

5.12.5.132 normalBuffers

`const CUdeviceptr* OptixBuildInputCurveArray::normalBuffers`

Reserved for future use.

5.12.5.133 normalStrideInBytes

`unsigned int OptixBuildInputCurveArray::normalStrideInBytes`

Reserved for future use.

5.12.5.134 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].

5.12.5.135 numBoundValues

`unsigned int OptixModuleCompileOptions::numBoundValues`

set to 0 if unused

5.12.5.136 numIndexTriplets

`unsigned int OptixBuildInputTriangleArray::numIndexTriplets`

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

5.12.5.137 numInstances [1/2]

`unsigned int OptixBuildInputInstanceArray::numInstances`

Number of elements in OptixBuildInputInstanceArray::instances.

5.12.5.138 numInstances [2/2]

`unsigned int OptixRelocateInputInstanceArray::numInstances`

Number of elements in OptixRelocateInputInstanceArray::traversableHandles. Must match OptixBuildInputInstanceArray::numInstances of the source build input.

5.12.5.139 numKeys

`unsigned short OptixMotionOptions::numKeys`

If numKeys > 1, motion is enabled. timeBegin, timeEnd and flags are all ignored when motion is disabled.

5.12.5.140 numMicromapHistogramEntries

```
unsigned int OptixOpacityMicromapArrayBuildInput
::numMicromapHistogramEntries
```

Number of OptixOpacityMicromapHistogramEntry.

5.12.5.141 numMicromapUsageCounts

```
unsigned int OptixBuildInputOpacityMicromap::numMicromapUsageCounts
```

Number of OptixOpacityMicromapUsageCount.

5.12.5.142 numPayloadTypes

```
unsigned int OptixModuleCompileOptions::numPayloadTypes
```

The number of different payload types available for compilation. Must be zero if OptixPipelineCompileOptions::numPayloadValues is not zero.

5.12.5.143 numPayloadValues [1/2]

```
unsigned int OptixPayloadType::numPayloadValues
```

The number of 32b words the payload of this type holds.

5.12.5.144 numPayloadValues [2/2]

```
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.

5.12.5.145 numPrimitives [1/2]

```
unsigned int OptixBuildInputCurveArray::numPrimitives
```

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

5.12.5.146 numPrimitives [2/2]

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

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

5.12.5.147 numSbtRecords [1/4]

```
unsigned int OptixBuildInputTriangleArray::numSbtRecords
```

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

5.12.5.148 numSbtRecords [2/4]

```
unsigned int OptixRelocateInputTriangleArray::numSbtRecords
```

Number of sbt records available to the sbt index offset override. Must match OptixBuildInputTriangleArray::numSbtRecords of the source build input.

5.12.5.149 numSbtRecords [3/4]

```
unsigned int OptixBuildInputSphereArray::numSbtRecords
```

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

5.12.5.150 numSbtRecords [4/4]

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

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

5.12.5.151 numVertices [1/3]

```
unsigned int OptixBuildInputTriangleArray::numVertices
```

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

5.12.5.152 numVertices [2/3]

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

Number of vertices in each buffer in vertexBuffers.

5.12.5.153 numVertices [3/3]

```
unsigned int OptixBuildInputSphereArray::numVertices
```

Number of vertices in each buffer in vertexBuffers.

5.12.5.154 opacityMicromap [1/2]

```
OptixBuildInputOpacityMicromap OptixBuildInputTriangleArray
::opacityMicromap
```

Optional opacity micromap inputs.

5.12.5.155 opacityMicromap [2/2]

```
OptixRelocateInputOpacityMicromap OptixRelocateInputTriangleArray
::opacityMicromap
```

Opacity micromap inputs.

5.12.5.156 opacityMicromapArray [1/2]

```
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.

5.12.5.157 opacityMicromapArray [2/2]

```
CUdeviceptr OptixRelocateInputOpacityMicromap::opacityMicromapArray
```

Device pointer to a reloated opacity micromap array used by the source build input array. May be zero when no micromaps where used in the source accel, or the referenced opacity micromaps don't require relocation (for example relocation of a GAS on the source device).

5.12.5.158 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.

5.12.5.159 optLevel

`OptixCompileOptimizationLevel OptixModuleCompileOptions::optLevel`

Optimization level. May vary within a pipeline.

5.12.5.160 output [1/2]

`CUdeviceptr OptixMicromapBuffers::output`

Output buffer.

5.12.5.161 output [2/2]

`OptixImage2D OptixDenoiserLayer::output`

5.12.5.162 outputInternalGuideLayer

`OptixImage2D OptixDenoiserGuideLayer::outputInternalGuideLayer`

5.12.5.163 outputSizeInBytes [1/3]

`size_t OptixMicromapBufferSizes::outputSizeInBytes`

5.12.5.164 outputSizeInBytes [2/3]

`size_t OptixMicromapBuffers::outputSizeInBytes`

Output buffer size.

5.12.5.165 outputSizeInBytes [3/3]

`size_t OptixAccelBufferSizes::outputSizeInBytes`

The size in bytes required for the outputBuffer parameter to optixAccelBuild when doing a build (OPTIX_BUILD_OPERATION_BUILD).

5.12.5.166 overlapWindowSizeInPixels

`unsigned int OptixDenoiserSizes::overlapWindowSizeInPixels`

Overlap on all four tile sides.

5.12.5.167 [1/6]

`char { ... } ::pad[1024]`

5.12.5.168 pad [2/6]

`char OptixBuildInput::pad[1024]`

5.12.5.169 pad [3/6]

`unsigned int OptixInstance::pad[2]`

round up to 80-byte, to ensure 16-byte alignment

5.12.5.170 pad [4/6]

`unsigned int OptixStaticTransform::pad[2]`

Padding to make the transformations 16 byte aligned.

5.12.5.171 pad [5/6]

`unsigned int OptixMatrixMotionTransform::pad[3]`

Padding to make the transformation 16 byte aligned.

5.12.5.172 pad [6/6]

`unsigned int OptixSRTMotionTransform::pad[3]`

Padding to make the SRT data 16 byte aligned.

5.12.5.173 payloadSemantics

`const unsigned int* OptixPayloadType::payloadSemantics`

Points to host array of payload word semantics, size must match numPayloadValues.

5.12.5.174 payloadType

`OptixPayloadType* OptixProgramGroupOptions::payloadType`

Specifies the payload type of this program group. All programs in the group must support the payload type (Program support for a type is specified by calling.

See also optixSetPayloadTypes or otherwise all types specified in

OptixModuleCompileOptions are supported). If a program is not available for the requested payload type, optixProgramGroupCreate returns OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH. If the payloadType is left zero, a unique type is deduced. The payload type can be uniquely deduced if there is exactly one payload type for which all programs in the group are available. If the payload type could not be deduced uniquely optixProgramGroupCreate returns OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED.

5.12.5.175 payloadTypes

`OptixPayloadType* OptixModuleCompileOptions::payloadTypes`

Points to host array of payload type definitions, size must match numPayloadTypes.

5.12.5.176 perMicromapDescBuffer

`CUdeviceptr OptixOpacityMicromapArrayBuildInput::perMicromapDescBuffer`

One OptixOpacityMicromapDesc entry per opacity micromap.

5.12.5.177 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).

5.12.5.178 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.

5.12.5.179 pipelineParamOffsetInBytes

`size_t OptixModuleCompileBoundValueEntry::pipelineParamOffsetInBytes`

5.12.5.180 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 at least OptixDenoiserSizes::internalGuideLayerSizeInBytes.

5.12.5.181 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.

5.12.5.182 previousOutput

`OptixImage2D OptixDenoiserLayer::previousOutput`

5.12.5.183 previousOutputInternalGuideLayer

`OptixImage2D OptixDenoiserGuideLayer::previousOutputInternalGuideLayer`

5.12.5.184 primitiveIndexOffset [1/4]

`unsigned int OptixBuildInputTriangleArray::primitiveIndexOffset`

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

5.12.5.185 primitiveIndexOffset [2/4]

`unsigned int OptixBuildInputCurveArray::primitiveIndexOffset`

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

5.12.5.186 primitiveIndexOffset [3/4]

`unsigned int OptixBuildInputSphereArray::primitiveIndexOffset`

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

5.12.5.187 primitiveIndexOffset [4/4]

`unsigned int OptixBuildInputCustomPrimitiveArray::primitiveIndexOffset`

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

5.12.5.188 pvx

`float OptixSRTData::pvx`

5.12.5.189 pvy

`float OptixSRTData::pvy`

5.12.5.190 pvz

`float OptixSRTData::pvz`

5.12.5.191 qw

`float OptixSRTData::qw`

5.12.5.192 qx

`float OptixSRTData::qx`

5.12.5.193 qy

`float OptixSRTData::qy`

5.12.5.194 qz

`float OptixSRTData::qz`

5.12.5.195 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.

5.12.5.196 radiusStrideInBytes

`unsigned int OptixBuildInputSphereArray::radiusStrideInBytes`

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

5.12.5.197 raygen [1/2]

`OptixProgramGroupSingleModule OptixProgramGroupDesc::raygen`

See also OPTIX_PROGRAM_GROUP_KIND_RAYGEN

5.12.5.198 [2/2]

`OptixProgramGroupSingleModule { ... } ::raygen`

See also OPTIX_PROGRAM_GROUP_KIND_RAYGEN

5.12.5.199 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.

5.12.5.200 result

`CUdeviceptr OptixAccelEmitDesc::result`

Output buffer for the properties.

5.12.5.201 rowStrideInBytes

`unsigned int OptixImage2D::rowStrideInBytes`

Stride between subsequent rows of the image (in bytes).

5.12.5.202 sbtIndexOffsetBuffer [1/3]

`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.

5.12.5.203 sbtIndexOffsetBuffer [2/3]

`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.

5.12.5.204 sbtIndexOffsetBuffer [3/3]

`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.

5.12.5.205 sbtIndexOffsetSizeInBytes [1/3]

`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).

5.12.5.206 sbtIndexOffsetSizeInBytes [2/3]

`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).

5.12.5.207 sbtIndexOffsetSizeInBytes [3/3]

`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).

5.12.5.208 sbtIndexOffsetStrideInBytes [1/3]

```
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).

5.12.5.209 sbtIndexOffsetStrideInBytes [2/3]

```
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).

5.12.5.210 sbtIndexOffsetStrideInBytes [3/3]

```
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).

5.12.5.211 sbtOffset

```
unsigned int OptixInstance::sbtOffset
```

SBT record offset. Will only be used for instances of geometry acceleration structure (GAS) objects. Needs to be set to 0 for instances of instance acceleration structure (IAS) objects. The maximal SBT offset can be queried using OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_SBT_OFFSET.

5.12.5.212 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.

5.12.5.213 sizeInBytes

```
size_t OptixModuleCompileBoundValueEntry::sizeInBytes
```

5.12.5.214 [1/2]

```
OptixBuildInputSphereArray { ... } ::sphereArray
```

Sphere inputs.

5.12.5.215 sphereArray [2/2]

```
OptixBuildInputSphereArray OptixBuildInput::sphereArray
```

Sphere inputs.

5.12.5.216 srtData

```
OptixSRTData OptixSRTMotionTransform::srtData[2]
```

The actual SRT data describing the transformation.

5.12.5.217 stateSizeInBytes

`size_t OptixDenoiserSizes::stateSizeInBytes`

Size of state memory passed to optixDenoiserSetup, optixDenoiserInvoke.

5.12.5.218 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.

5.12.5.219 subdivisionLevel [1/3]

`unsigned int OptixOpacityMicromapUsageCount::subdivisionLevel`

Number of micro-triangles is 4^level. Valid levels are [0, 12].

5.12.5.220 subdivisionLevel [2/3]

`unsigned short OptixOpacityMicromapDesc::subdivisionLevel`

Number of micro-triangles is 4^level. Valid levels are [0, 12].

5.12.5.221 subdivisionLevel [3/3]

`unsigned int OptixOpacityMicromapHistogramEntry::subdivisionLevel`

Number of micro-triangles is 4^level. Valid levels are [0, 12].

5.12.5.222 sx

`float OptixSRTData::sx`

5.12.5.223 sy

`float OptixSRTData::sy`

5.12.5.224 sz

`float OptixSRTData::sz`

5.12.5.225 temp

`CUdeviceptr OptixMicromapBuffers::temp`

Temp buffer.

5.12.5.226 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.

5.12.5.227 tempSizeInBytes [1/3]

```
size_t OptixMicromapBufferSizes::tempSizeInBytes
```

5.12.5.228 tempSizeInBytes [2/3]

```
size_t OptixMicromapBuffers::tempSizeInBytes
```

Temp buffer size.

5.12.5.229 tempSizeInBytes [3/3]

```
size_t OptixAccelBufferSizes::tempSizeInBytes
```

The size in bytes required for the tempBuffer paramter to optixAccelBuild when doing a build (OPTIX_BUILD_OPERATION_BUILD).

5.12.5.230 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.

5.12.5.231 timeBegin

```
float OptixMotionOptions::timeBegin
```

Point in time where motion starts. Must be lesser than timeEnd.

5.12.5.232 timeEnd

```
float OptixMotionOptions::timeEnd
```

Point in time where motion ends. Must be greater than timeBegin.

5.12.5.233 transform [1/3]

```
float OptixInstance::transform[12]
```

affine object-to-world transformation as 3x4 matrix in row-major layout

5.12.5.234 transform [2/3]

```
float OptixStaticTransform::transform[12]
```

Affine object-to-world transformation as 3x4 matrix in row-major layout.

5.12.5.235 transform [3/3]

```
float OptixMatrixMotionTransform::transform[2][12]
```

Affine object-to-world transformation as 3x4 matrix in row-major layout.

5.12.5.236 transformFormat

```
OptixTransformFormat OptixBuildInputTriangleArray::transformFormat
```

See also OptixTransformFormat

5.12.5.237 traversableGraphFlags

`unsigned int OptixPipelineCompileOptions::traversableGraphFlags`

Traversable graph bitfield. See OptixTraversableGraphFlags.

5.12.5.238 traversableHandle

`OptixTraversableHandle OptixInstance::traversableHandle`

Set with an OptixTraversableHandle.

5.12.5.239 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).

5.12.5.240 [1/4]

`OptixBuildInputTriangleArray { ... } ::triangleArray`

Triangle inputs.

5.12.5.241 triangleArray [2/4]

`OptixBuildInputTriangleArray OptixBuildInput::triangleArray`

Triangle inputs.

5.12.5.242 [3/4]

`OptixRelocateInputTriangleArray { ... } ::triangleArray`

Triangle inputs.

5.12.5.243 triangleArray [4/4]

`OptixRelocateInputTriangleArray OptixRelocateInput::triangleArray`

Triangle inputs.

5.12.5.244 tx

`float OptixSRTData::tx`

5.12.5.245 ty

`float OptixSRTData::ty`

5.12.5.246 type [1/3]

`OptixBuildInputType OptixBuildInput::type`

The type of the build input.

5.12.5.247 type [2/3]

`OptixBuildInputType OptixRelocateInput::type`

The type of the build input to relocate.

5.12.5.248 type [3/3]

`OptixAccelPropertyType OptixAccelEmitDesc::type`

Requested property.

5.12.5.249 tz

`float OptixSRTData::tz`

5.12.5.250 usesMotionBlur [1/2]

`int OptixPipelineCompileOptions::usesMotionBlur`

Boolean value indicating whether motion blur could be used.

5.12.5.251 usesMotionBlur [2/2]

`int OptixBuiltinISOptions::usesMotionBlur`

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

5.12.5.252 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.

5.12.5.253 validationMode

`OptixDeviceContextValidationMode OptixDeviceContextOptions::validationMode`

Validation mode of context.

5.12.5.254 vertexBuffers [1/3]

`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.

5.12.5.255 vertexBuffers [2/3]

`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).

5.12.5.256 vertexBuffers [3/3]

```
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).

5.12.5.257 vertexFormat

```
OptixVertexFormat OptixBuildInputTriangleArray::vertexFormat
```

See also OptixVertexFormat

5.12.5.258 vertexStrideInBytes [1/3]

```
unsigned int OptixBuildInputTriangleArray::vertexStrideInBytes
```

Stride between vertices. If set to zero, vertices are assumed to be tightly packed and stride is inferred from vertexFormat.

5.12.5.259 vertexStrideInBytes [2/3]

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

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

5.12.5.260 vertexStrideInBytes [3/3]

```
unsigned int OptixBuildInputSphereArray::vertexStrideInBytes
```

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

5.12.5.261 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.

5.12.5.262 width

```
unsigned int OptixImage2D::width
```

Width of the image (in pixels)

5.12.5.263 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.

5.12.5.264 widthStrideInBytes

`unsigned int OptixBuildInputCurveArray::widthStrideInBytes`

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

5.12.5.265 withoutOverlapScratchSizeInBytes

`size_t OptixDenoiserSizes::withoutOverlapScratchSizeInBytes`

Size of scratch memory passed to optixDenoiserSetup, optixDenoiserInvoke. No overlap added.

5.12.5.266 withOverlapScratchSizeInBytes

`size_t OptixDenoiserSizes::withOverlapScratchSizeInBytes`

Size of scratch memory passed to optixDenoiserSetup, optixDenoiserInvoke. Overlap added to dimensions passed to optixDenoiserComputeMemoryResources.

5.13 Function Table

Classes

- struct OptixFunctionTable

Typedefs

- typedef struct OptixFunctionTable OptixFunctionTable

Variables

- OptixFunctionTable g_optixFunctionTable

Error handling

- const char *(* OptixFunctionTable::optixGetErrorName)(OptixResult result)
- const char *(* OptixFunctionTable::optixGetErrorString)(OptixResult result)

Device context

- OptixResult(* OptixFunctionTable::optixDeviceContextCreate)(CUcontext fromContext, const OptixDeviceContextOptions *options, OptixDeviceContext *context)
- OptixResult(* OptixFunctionTable::optixDeviceContextDestroy)(OptixDeviceContext context)
- OptixResult(* OptixFunctionTable::optixDeviceContextGetProperty)(OptixDeviceContext context, OptixDeviceProperty property, void *value, size_t sizeInBytes)
- OptixResult(* OptixFunctionTable::optixDeviceContextSetLogCallback)(OptixDeviceContext context, OptixLogCallback callbackFunction, void *callbackData, unsigned int callbackLevel)
- OptixResult(* OptixFunctionTable::optixDeviceContextSetCacheEnabled)(OptixDeviceContext context, int enabled)
- OptixResult(* OptixFunctionTable::optixDeviceContextSetCacheLocation)(OptixDeviceContext context, const char *location)
- OptixResult(* OptixFunctionTable ::optixDeviceContextSetCacheDatabaseSizes)(OptixDeviceContext context, size_t lowWaterMark, size_t highWaterMark)
- OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheEnabled)(OptixDeviceContext context, int *enabled)
- OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheLocation)(OptixDeviceContext context, char *location, size_t locationSize)

- OptixResult(∗ OptixFunctionTable ::optixDeviceContextGetCacheDatabaseSizes)(OptixDeviceContext context, size_t ∗lowWaterMark, size_t ∗highWaterMark)

Modules

- OptixResult(∗ OptixFunctionTable::optixModuleCreateFromPTX)(OptixDeviceContext context, const OptixModuleCompileOptions ∗moduleCompileOptions, const OptixPipelineCompileOptions ∗pipelineCompileOptions, const char ∗PTX, size_t PTXsize, char ∗logString, size_t ∗logStringSize, OptixModule ∗module)
- OptixResult(∗ OptixFunctionTable::optixModuleCreateFromPTXWithTasks)(OptixDeviceContext context, const OptixModuleCompileOptions ∗moduleCompileOptions, const OptixPipelineCompileOptions ∗pipelineCompileOptions, const char ∗PTX, size_t PTXsize, char ∗logString, size_t ∗logStringSize, OptixModule ∗module, OptixTask ∗firstTask)
- OptixResult(∗ OptixFunctionTable::optixModuleGetCompilationState)(OptixModule module, OptixModuleCompileState ∗state)
- OptixResult(∗ OptixFunctionTable::optixModuleDestroy)(OptixModule module)
- OptixResult(∗ OptixFunctionTable::optixBuiltinISModuleGet)(OptixDeviceContext context, const OptixModuleCompileOptions ∗moduleCompileOptions, const OptixPipelineCompileOptions ∗pipelineCompileOptions, const OptixBuiltinISOptions ∗builtinISOptions, OptixModule ∗builtinModule)

Tasks

- OptixResult(∗ OptixFunctionTable::optixTaskExecute)(OptixTask task, OptixTask ∗additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int ∗numAdditionalTasksCreated)

Program groups

- OptixResult(∗ OptixFunctionTable::optixProgramGroupCreate)(OptixDeviceContext context, const OptixProgramGroupDesc ∗programDescriptions, unsigned int numProgramGroups, const OptixProgramGroupOptions ∗options, char ∗logString, size_t ∗logStringSize, OptixProgramGroup ∗programGroups)
- OptixResult(∗ OptixFunctionTable::optixProgramGroupDestroy)(OptixProgramGroup programGroup)
- OptixResult(∗ OptixFunctionTable::optixProgramGroupGetStackSize)(OptixProgramGroup programGroup, OptixStackSizes ∗stackSizes)

Pipeline

- OptixResult(∗ OptixFunctionTable::optixPipelineCreate)(OptixDeviceContext context, const OptixPipelineCompileOptions ∗pipelineCompileOptions, const OptixPipelineLinkOptions ∗pipelineLinkOptions, const OptixProgramGroup ∗programGroups, unsigned int numProgramGroups, char ∗logString, size_t ∗logStringSize, OptixPipeline ∗pipeline)
- OptixResult(∗ OptixFunctionTable::optixPipelineDestroy)(OptixPipeline pipeline)
- OptixResult(∗ OptixFunctionTable::optixPipelineSetStackSize)(OptixPipeline pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)

Acceleration structures

- OptixResult(∗ OptixFunctionTable::optixAccelComputeMemoryUsage)(OptixDeviceContext context, const OptixAccelBuildOptions ∗accelOptions, const OptixBuildInput ∗buildInputs, unsigned int numBuildInputs, OptixAccelBufferSizes ∗bufferSizes)

- 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)
- OptixResult(∗ OptixFunctionTable::optixAccelGetRelocationInfo)(OptixDeviceContext context, OptixTraversableHandle handle, OptixRelocationInfo ∗info)
- OptixResult(∗ OptixFunctionTable::optixCheckRelocationCompatibility)(OptixDeviceContext context, const OptixRelocationInfo ∗info, int ∗compatible)
- OptixResult(∗ OptixFunctionTable::optixAccelRelocate)(OptixDeviceContext context, CUstream stream, const OptixRelocationInfo ∗info, const OptixRelocateInput ∗relocateInputs, size_t numRelocateInputs, CUdeviceptr targetAccel, size_t targetAccelSizeInBytes, OptixTraversableHandle ∗targetHandle)
- OptixResult(∗ OptixFunctionTable::optixAccelCompact)(OptixDeviceContext context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle ∗outputHandle)
- OptixResult(∗ OptixFunctionTable ::optixConvertPointerToTraversableHandle)(OptixDeviceContext onDevice, CUdeviceptr pointer, OptixTraversableType traversableType, OptixTraversableHandle ∗traversableHandle)
- OptixResult(∗ OptixFunctionTable ::optixOpacityMicromapArrayComputeMemoryUsage)(OptixDeviceContext context, const OptixOpacityMicromapArrayBuildInput ∗buildInput, OptixMicromapBufferSizes ∗bufferSizes)
- OptixResult(∗ OptixFunctionTable::optixOpacityMicromapArrayBuild)(OptixDeviceContext context, CUstream stream, const OptixOpacityMicromapArrayBuildInput ∗buildInput, const OptixMicromapBuffers ∗buffers)
- OptixResult(∗ OptixFunctionTable ::optixOpacityMicromapArrayGetRelocationInfo)(OptixDeviceContext context, CUdeviceptr opacityMicromapArray, OptixRelocationInfo ∗info)
- OptixResult(∗ OptixFunctionTable::optixOpacityMicromapArrayRelocate)(OptixDeviceContext context, CUstream stream, const OptixRelocationInfo ∗info, CUdeviceptr targetOpacityMicromapArray, size_t targetOpacityMicromapArraySizeInBytes)
- void(∗ OptixFunctionTable::reserved1)(void)
- void(∗ OptixFunctionTable::reserved2)(void)

Launch

- OptixResult(∗ OptixFunctionTable::optixSbtRecordPackHeader)(OptixProgramGroup programGroup, void ∗sbtRecordHeaderHostPointer)
- OptixResult(∗ OptixFunctionTable::optixLaunch)(OptixPipeline pipeline, CUstream stream, CUdeviceptr pipelineParams, size_t pipelineParamsSize, const OptixShaderBindingTable ∗sbt, unsigned int width, unsigned int height, unsigned int depth)

Denoiser

- OptixResult(∗ OptixFunctionTable::optixDenoiserCreate)(OptixDeviceContext context, OptixDenoiserModelKind modelKind, const OptixDenoiserOptions ∗options, OptixDenoiser ∗returnHandle)
- OptixResult(∗ OptixFunctionTable::optixDenoiserDestroy)(OptixDenoiser handle)
- OptixResult(∗ OptixFunctionTable::optixDenoiserComputeMemoryResources)(const OptixDenoiser handle, unsigned int maximumInputWidth, unsigned int maximumInputHeight, OptixDenoiserSizes ∗returnSizes)

- OptixResult(∗ OptixFunctionTable::optixDenoiserSetup)(OptixDenoiser denoiser, CUstream stream, unsigned int inputWidth, unsigned int inputHeight, CUdeviceptr state, size_t stateSizeInBytes, CUdeviceptr scratch, size_t scratchSizeInBytes)
- 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)
- OptixResult(∗ OptixFunctionTable::optixDenoiserComputeIntensity)(OptixDenoiser handle, CUstream stream, const OptixImage2D ∗inputImage, CUdeviceptr outputIntensity, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OptixResult(∗ OptixFunctionTable::optixDenoiserComputeAverageColor)(OptixDenoiser handle, CUstream stream, const OptixImage2D ∗inputImage, CUdeviceptr outputAverageColor, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OptixResult(∗ OptixFunctionTable::optixDenoiserCreateWithUserModel)(OptixDeviceContext context, const void ∗data, size_t dataSizeInBytes, OptixDenoiser ∗returnHandle)

5.13.1 Detailed Description

OptiX Function Table.

5.13.2 Typedef Documentation

5.13.2.1 OptixFunctionTable

```
typedef struct OptixFunctionTable OptixFunctionTable
```

The function table containing all API functions.

See optixInit( ) and optixInitWithHandle( ).

5.13.3 Variable Documentation

5.13.3.1 g_optixFunctionTable

```
OptixFunctionTable g_optixFunctionTable
```

If the stubs in optix_stubs.h are used, then the function table needs to be defined in exactly one translation unit. This can be achieved by including this header file in that translation unit.

5.13.3.2 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( ).

5.13.3.3 optixAccelCompact

```
OptixResult(∗ OptixFunctionTable::optixAccelCompact) (OptixDeviceContext
context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr
outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle
∗outputHandle)
```

See optixAccelCompact( ).

5.13.3.4 optixAccelComputeMemoryUsage

```
OptixResult(* OptixFunctionTable::optixAccelComputeMemoryUsage)
(OptixDeviceContext context, const OptixAccelBuildOptions *accelOptions,
const OptixBuildInput *buildInputs, unsigned int numBuildInputs,
OptixAccelBufferSizes *bufferSizes)
```

See optixAccelComputeMemoryUsage( ).

5.13.3.5 optixAccelGetRelocationInfo

```
OptixResult(* OptixFunctionTable::optixAccelGetRelocationInfo)
(OptixDeviceContext context, OptixTraversableHandle handle,
OptixRelocationInfo *info)
```

See optixAccelGetRelocationInfo( ).

5.13.3.6 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( ).

5.13.3.7 optixBuiltinISModuleGet

```
OptixResult(* OptixFunctionTable::optixBuiltinISModuleGet)
(OptixDeviceContext context, const OptixModuleCompileOptions
*moduleCompileOptions, const OptixPipelineCompileOptions
*pipelineCompileOptions, const OptixBuiltinISOptions *builtinISOptions,
OptixModule *builtinModule)
```

See optixBuiltinISModuleGet( ).

5.13.3.8 optixCheckRelocationCompatibility

```
OptixResult(* OptixFunctionTable::optixCheckRelocationCompatibility)
(OptixDeviceContext context, const OptixRelocationInfo *info, int
*compatible)
```

See optixCheckRelocationCompatibility( ).

5.13.3.9 optixConvertPointerToTraversableHandle

```
OptixResult(* OptixFunctionTable::optixConvertPointerToTraversableHandle)
(OptixDeviceContext onDevice, CUdeviceptr pointer, OptixTraversableType
traversableType, OptixTraversableHandle *traversableHandle)
```

See optixConvertPointerToTraversableHandle( ).

5.13.3.10 optixDenoiserComputeAverageColor

```
OptixResult(* OptixFunctionTable::optixDenoiserComputeAverageColor)
```

```
(OptixDenoiser handle, CUstream stream, const OptixImage2D *inputImage,
CUdeviceptr outputAverageColor, CUdeviceptr scratch, size_t
scratchSizeInBytes)
```

See optixDenoiserComputeAverageColor( ).

5.13.3.11 optixDenoiserComputeIntensity

```
OptixResult(* OptixFunctionTable::optixDenoiserComputeIntensity)
(OptixDenoiser handle, CUstream stream, const OptixImage2D *inputImage,
CUdeviceptr outputIntensity, CUdeviceptr scratch, size_t scratchSizeInBytes)
```

See optixDenoiserComputeIntensity( ).

5.13.3.12 optixDenoiserComputeMemoryResources

```
OptixResult(* OptixFunctionTable::optixDenoiserComputeMemoryResources)
(const OptixDenoiser handle, unsigned int maximumInputWidth, unsigned int
maximumInputHeight, OptixDenoiserSizes *returnSizes)
```

See optixDenoiserComputeMemoryResources( ).

5.13.3.13 optixDenoiserCreate

```
OptixResult(* OptixFunctionTable::optixDenoiserCreate) (OptixDeviceContext
context, OptixDenoiserModelKind modelKind, const OptixDenoiserOptions
*options, OptixDenoiser *returnHandle)
```

See optixDenoiserCreate( ).

5.13.3.14 optixDenoiserCreateWithUserModel

```
OptixResult(* OptixFunctionTable::optixDenoiserCreateWithUserModel)
(OptixDeviceContext context, const void *data, size_t dataSizeInBytes,
OptixDenoiser *returnHandle)
```

See optixDenoiserCreateWithUserModel( ).

5.13.3.15 optixDenoiserDestroy

```
OptixResult(* OptixFunctionTable::optixDenoiserDestroy) (OptixDenoiser
handle)
```

See optixDenoiserDestroy( ).

5.13.3.16 optixDenoiserInvoke

```
OptixResult(* OptixFunctionTable::optixDenoiserInvoke) (OptixDenoiser
denoiser, CUstream stream, const OptixDenoiserParams *params, CUdeviceptr
denoiserState, size_t denoiserStateSizeInBytes, const
OptixDenoiserGuideLayer *guideLayer, const OptixDenoiserLayer *layers,
unsigned int numLayers, unsigned int inputOffsetX, unsigned int
inputOffsetY, CUdeviceptr scratch, size_t scratchSizeInBytes)
```

See optixDenoiserInvoke( ).

5.13.3.17 optixDenoiserSetup

```
OptixResult(* OptixFunctionTable::optixDenoiserSetup) (OptixDenoiser
```

```
denoiser, CUstream stream, unsigned int inputWidth, unsigned int
inputHeight, CUdeviceptr state, size_t stateSizeInBytes, CUdeviceptr
scratch, size_t scratchSizeInBytes)
```

See optixDenoiserSetup( ).

5.13.3.18 optixDeviceContextCreate

```
OptixResult(* OptixFunctionTable::optixDeviceContextCreate) (CUcontext
fromContext, const OptixDeviceContextOptions *options, OptixDeviceContext
*context)
```

See optixDeviceContextCreate( ).

5.13.3.19 optixDeviceContextDestroy

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

See optixDeviceContextDestroy( ).

5.13.3.20 optixDeviceContextGetCacheDatabaseSizes

```
OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheDatabaseSizes)
(OptixDeviceContext context, size_t *lowWaterMark, size_t *highWaterMark)
```

See optixDeviceContextGetCacheDatabaseSizes( ).

5.13.3.21 optixDeviceContextGetCacheEnabled

```
OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheEnabled)
(OptixDeviceContext context, int *enabled)
```

See optixDeviceContextGetCacheEnabled( ).

5.13.3.22 optixDeviceContextGetCacheLocation

```
OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheLocation)
(OptixDeviceContext context, char *location, size_t locationSize)
```

See optixDeviceContextGetCacheLocation( ).

5.13.3.23 optixDeviceContextGetProperty

```
OptixResult(* OptixFunctionTable::optixDeviceContextGetProperty)
(OptixDeviceContext context, OptixDeviceProperty property, void *value, size
_t sizeInBytes)
```

See optixDeviceContextGetProperty( ).

5.13.3.24 optixDeviceContextSetCacheDatabaseSizes

```
OptixResult(* OptixFunctionTable::optixDeviceContextSetCacheDatabaseSizes)
(OptixDeviceContext context, size_t lowWaterMark, size_t highWaterMark)
```

See optixDeviceContextSetCacheDatabaseSizes( ).

5.13.3.25 optixDeviceContextSetCacheEnabled

```
OptixResult(* OptixFunctionTable::optixDeviceContextSetCacheEnabled)
```

```
(OptixDeviceContext context, int enabled)
```

See optixDeviceContextSetCacheEnabled( ).

5.13.3.26 optixDeviceContextSetCacheLocation

```
OptixResult(* OptixFunctionTable::optixDeviceContextSetCacheLocation)
(OptixDeviceContext context, const char *location)
```

See optixDeviceContextSetCacheLocation( ).

5.13.3.27 optixDeviceContextSetLogCallback

```
OptixResult(* OptixFunctionTable::optixDeviceContextSetLogCallback)
(OptixDeviceContext context, OptixLogCallback callbackFunction, void
*callbackData, unsigned int callbackLevel)
```

See optixDeviceContextSetLogCallback( ).

5.13.3.28 optixGetErrorName

```
const char *(* OptixFunctionTable::optixGetErrorName) (OptixResult result)
```

See optixGetErrorName( ).

5.13.3.29 optixGetErrorString

```
const char *(* OptixFunctionTable::optixGetErrorString) (OptixResult result)
```

See optixGetErrorString( ).

5.13.3.30 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( ).

5.13.3.31 optixModuleCreateFromPTX

```
OptixResult(* OptixFunctionTable::optixModuleCreateFromPTX)
(OptixDeviceContext context, const OptixModuleCompileOptions
*moduleCompileOptions, const OptixPipelineCompileOptions
*pipelineCompileOptions, const char *PTX, size_t PTXsize, char *logString,
size_t *logStringSize, OptixModule *module)
```

See optixModuleCreateFromPTX( ).

5.13.3.32 optixModuleCreateFromPTXWithTasks

```
OptixResult(* OptixFunctionTable::optixModuleCreateFromPTXWithTasks)
(OptixDeviceContext context, const OptixModuleCompileOptions
*moduleCompileOptions, const OptixPipelineCompileOptions
*pipelineCompileOptions, const char *PTX, size_t PTXsize, char *logString,
size_t *logStringSize, OptixModule *module, OptixTask *firstTask)
```

See optixModuleCreateFromPTXWithTasks( ).

5.13.3.33 optixModuleDestroy

```
OptixResult(* OptixFunctionTable::optixModuleDestroy) (OptixModule module)
```

See optixModuleDestroy( ).

5.13.3.34 optixModuleGetCompilationState

```
OptixResult(* OptixFunctionTable::optixModuleGetCompilationState)
(OptixModule module, OptixModuleCompileState *state)
```

See optixModuleGetCompilationState( ).

5.13.3.35 optixOpacityMicromapArrayBuild

```
OptixResult(* OptixFunctionTable::optixOpacityMicromapArrayBuild)
(OptixDeviceContext context, CUstream stream, const
OptixOpacityMicromapArrayBuildInput *buildInput, const OptixMicromapBuffers
*buffers)
```

See optixOpacityMicromapArrayBuild( ).

5.13.3.36 optixOpacityMicromapArrayComputeMemoryUsage

```
OptixResult(* OptixFunctionTable
::optixOpacityMicromapArrayComputeMemoryUsage) (OptixDeviceContext context,
const OptixOpacityMicromapArrayBuildInput *buildInput,
OptixMicromapBufferSizes *bufferSizes)
```

See optixOpacityMicromapArrayComputeMemoryUsage( ).

5.13.3.37 optixOpacityMicromapArrayGetRelocationInfo

```
OptixResult(* OptixFunctionTable
::optixOpacityMicromapArrayGetRelocationInfo) (OptixDeviceContext context,
CUdeviceptr opacityMicromapArray, OptixRelocationInfo *info)
```

See optixOpacityMicromapArrayGetRelocationInfo( ).

5.13.3.38 optixOpacityMicromapArrayRelocate

```
OptixResult(* OptixFunctionTable::optixOpacityMicromapArrayRelocate)
(OptixDeviceContext context, CUstream stream, const OptixRelocationInfo
*info, CUdeviceptr targetOpacityMicromapArray, size_t
targetOpacityMicromapArraySizeInBytes)
```

See optixOpacityMicromapArrayRelocate( ).

5.13.3.39 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( ).

5.13.3.40 optixPipelineDestroy

```
OptixResult(* OptixFunctionTable::optixPipelineDestroy) (OptixPipeline
pipeline)
```

See optixPipelineDestroy( ).

5.13.3.41 optixPipelineSetStackSize

```
OptixResult(* OptixFunctionTable::optixPipelineSetStackSize) (OptixPipeline
pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int
directCallableStackSizeFromState, unsigned int continuationStackSize,
unsigned int maxTraversableGraphDepth)
```

See optixPipelineSetStackSize( ).

5.13.3.42 optixProgramGroupCreate

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

See optixProgramGroupCreate( ).

5.13.3.43 optixProgramGroupDestroy

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

See optixProgramGroupDestroy( ).

5.13.3.44 optixProgramGroupGetStackSize

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

See optixProgramGroupGetStackSize( ).

5.13.3.45 optixSbtRecordPackHeader

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

See optixConvertPointerToTraversableHandle( ).

5.13.3.46 optixTaskExecute

```
OptixResult(* OptixFunctionTable::optixTaskExecute) (OptixTask task,
OptixTask *additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int
*numAdditionalTasksCreated)
```

See optixTaskExecute( ).

5.13.3.47 reserved1

```
void(* OptixFunctionTable::reserved1) (void)
```

See optixAccelComputeMemoryUsage( ).

5.13.3.48 reserved2

```
void(* OptixFunctionTable::reserved2) (void)
```

See optixAccelComputeMemoryUsage( ).

5.14 Utilities

Classes

- struct OptixUtilDenoiserImageTile

Functions

- OptixResult optixUtilAccumulateStackSizes (OptixProgramGroup programGroup, OptixStackSizes *stackSizes)
- 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)
- 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 optixInitWithHandle (void **handlePtr)
- OptixResult optixInit (void)
- OptixResult optixUninitWithHandle (void *handle)

Variables

- OptixImage2D OptixUtilDenoiserImageTile::input
- OptixImage2D OptixUtilDenoiserImageTile::output
- unsigned int OptixUtilDenoiserImageTile::inputOffsetX
- unsigned int OptixUtilDenoiserImageTile::inputOffsetY

5.14.1 Detailed Description

OptiX Utilities.

5.14.2 Function Documentation

5.14.2.1 optixInit()

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

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

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

5.14.2.2 optixInitWithHandle()

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

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

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

See also optixUninitWithHandle

5.14.2.3 optixUninitWithHandle()

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

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

See also optixInitWithHandle

5.14.2.4 optixUtilAccumulateStackSizes()

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

Retrieves direct and continuation stack sizes for each program in the program group and accumulates the upper bounds in the correponding 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.

5.14.2.5 optixUtilComputeStackSizes()

```
OptixResult optixUtilComputeStackSizes (
          const OptixStackSizes * stackSizes,
          unsigned int maxTraceDepth,
          unsigned int maxCCDepth,
          unsigned int maxDCDepth,
          unsigned int * directCallableStackSizeFromTraversal,
```

```
        unsigned int * directCallableStackSizeFromState,
        unsigned int * continuationStackSize )  [inline]
```

Computes the stack size values needed to configure a pipeline.

See the programming guide for an explanation of the formula.

Parameters

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

5.14.2.6 optixUtilComputeStackSizesCssCCTree( )

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

Computes the stack size values needed to configure a pipeline.

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

See programming guide for an explanation of the formula.

Parameters

in	*stackSizes*	Accumulated stack sizes of all programs in the call graph.
in	*cssCCTree*	Maximum stack size used by calls trees of continuation callables.
in	*maxTraceDepth*	Maximum depth of optixTrace( ) calls.
in	*maxDCDepth*	Maximum depth of calls trees of direct callables.
out	*directCallableStackSizeFromTraversal*	Direct stack size requirement for direct callables invoked from IS or AH.
out	*directCallableStackSizeFromState*	Direct stack size requirement for direct callables invoked from RG, MS, or CH.

Parameters

out	*continuationStackSize*	Continuation stack requirement.

5.14.2.7 optixUtilComputeStackSizesDCSplit()

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

Computes the stack size values needed to configure a pipeline.

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

See programming guide for an explanation of the formula.

Parameters

in	*stackSizes*	Accumulated stack sizes of all programs in the call graph.
in	*dssDCFromTraversal*	Accumulated direct stack size of all DC programs invoked from IS or AH.
in	*dssDCFromState*	Accumulated direct stack size of all DC programs invoked from RG, MS, or CH.
in	*maxTraceDepth*	Maximum depth of optixTrace( ) calls.
in	*maxCCDepth*	Maximum depth of calls trees of continuation callables.
in	*maxDCDepthFromTraversal*	Maximum depth of calls trees of direct callables invoked from IS or AH.
in	*maxDCDepthFromState*	Maximum depth of calls trees of direct callables invoked from RG, MS, or CH.
out	*directCallableStackSizeFromTraversal*	Direct stack size requirement for direct callables invoked from IS or AH.
out	*directCallableStackSizeFromState*	Direct stack size requirement for direct callables invoked from RG, MS, or CH.
out	*continuationStackSize*	Continuation stack requirement.

5.14.2.8 optixUtilComputeStackSizesSimplePathTracer()

```
OptixResult optixUtilComputeStackSizesSimplePathTracer (
```

```
            OptixProgramGroup programGroupRG,
            OptixProgramGroup programGroupMS1,
            const OptixProgramGroup * programGroupCH1,
            unsigned int programGroupCH1Count,
            OptixProgramGroup programGroupMS2,
            const OptixProgramGroup * programGroupCH2,
            unsigned int programGroupCH2Count,
            unsigned int * directCallableStackSizeFromTraversal,
            unsigned int * directCallableStackSizeFromState,
            unsigned int * continuationStackSize )  [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.

5.14.2.9 optixUtilDenoiserInvokeTiled()

```
OptixResult optixUtilDenoiserInvokeTiled (
            OptixDenoiser denoiser,
            CUstream stream,
            const OptixDenoiserParams * params,
            CUdeviceptr denoiserState,
            size_t denoiserStateSizeInBytes,
            const OptixDenoiserGuideLayer * guideLayer,
            const OptixDenoiserLayer * layers,
            unsigned int numLayers,
            CUdeviceptr scratch,
            size_t scratchSizeInBytes,
            unsigned int overlapWindowSizeInPixels,
            unsigned int tileWidth,
            unsigned int tileHeight )  [inline]
```

Run denoiser on input layers see optixDenoiserInvoke additional parameters:

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

Parameters

in	*denoiser*
in	*stream*
in	*params*
in	*denoiserState*
in	*denoiserStateSizeInBytes*
in	*guideLayer*
in	*layers*
in	*numLayers*
in	*scratch*
in	*scratchSizeInBytes*
in	*overlapWindowSizeInPixels*
in	*tileWidth*
in	*tileHeight*

5.14.2.10 optixUtilDenoiserSplitImage()

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

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

Parameters

in	*input*	full resolution input image to be split
in	*output*	full resolution output image
in	*overlapWindowSizeInPixels*	see OptixDenoiserSizes, optixDenoiserComputeMemoryResources
in	*tileWidth*	maximum width of tiles
in	*tileHeight*	maximum height of tiles
out	*tiles*	list of tiles covering the input image

5.14.2.11 optixUtilGetPixelStride()

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

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

Parameters

in	*image*	Image containing the pixel stride

5.14.3 Variable Documentation

5.14.3.1 input

```
OptixImage2D OptixUtilDenoiserImageTile::input
```

5.14.3.2 inputOffsetX

```
unsigned int OptixUtilDenoiserImageTile::inputOffsetX
```

5.14.3.3 inputOffsetY

```
unsigned int OptixUtilDenoiserImageTile::inputOffsetY
```

5.14.3.4 output

```
OptixImage2D OptixUtilDenoiserImageTile::output
```

6 Namespace Documentation

6.1 optix_impl Namespace Reference

Functions

- static __forceinline__ __device__ void optixDumpStaticTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optixDumpMotionMatrixTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optixDumpSrtMatrixTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optixDumpInstanceFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optixDumpTransform (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optixDumpTransformList ()
- static __forceinline__ __device__ void optixDumpExceptionDetails ()
- static __forceinline__ __device__ float4 optixAddFloat4 (const float4 &a, const float4 &b)
- static __forceinline__ __device__ float4 optixMulFloat4 (const float4 &a, float b)
- static __forceinline__ __device__ uint4 optixLdg (unsigned long long addr)
- template<class T >
  static __forceinline__ __device__ T optixLoadReadOnlyAlign16 (const T *ptr)
- static __forceinline__ __device__ float4 optixMultiplyRowMatrix (const float4 vec, const float4 m0, const float4 m1, const float4 m2)
- static __forceinline__ __device__ void optixGetMatrixFromSrt (float4 &m0, float4 &m1, float4 &m2, const OptixSRTData &srt)
- static __forceinline__ __device__ void optixInvertMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ void optixLoadInterpolatedMatrixKey (float4 &m0, float4 &m1, float4 &m2, const float4 *matrix, const float t1)
- static __forceinline__ __device__ void optixLoadInterpolatedSrtKey (float4 &srt0, float4 &srt1, float4 &srt2, float4 &srt3, const float4 *srt, const float t1)

- static __forceinline__ __device__ void optixResolveMotionKey (float &localt, int &key, const OptixMotionOptions &options, const float globalt)
- static __forceinline__ __device__ void optixGetInterpolatedTransformation (float4 &trf0, float4 &trf1, float4 &trf2, const OptixMatrixMotionTransform *transformData, const float time)
- static __forceinline__ __device__ void optixGetInterpolatedTransformation (float4 &trf0, float4 &trf1, float4 &trf2, const OptixSRTMotionTransform *transformData, const float time)
- static __forceinline__ __device__ void optixGetInterpolatedTransformationFromHandle (float4 &trf0, float4 &trf1, float4 &trf2, const OptixTraversableHandle handle, const float time, const bool objectToWorld)
- static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ float3 optixTransformPoint (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &p)
- static __forceinline__ __device__ float3 optixTransformVector (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &v)
- static __forceinline__ __device__ float3 optixTransformNormal (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &n)

6.1.1 Function Documentation

6.1.1.1 optixAddFloat4()

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

6.1.1.2 optixDumpExceptionDetails()

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

6.1.1.3 optixDumpInstanceFromHandle()

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

6.1.1.4 optixDumpMotionMatrixTransformFromHandle()

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

6.1.1.5 optixDumpSrtMatrixTransformFromHandle()

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

6.1.1.6 optixDumpStaticTransformFromHandle()

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

6.1.1.7 optixDumpTransform()

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

6.1.1.8 optixDumpTransformList()

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

6.1.1.9 optixGetInterpolatedTransformation() [1/2]

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

6.1.1.10 optixGetInterpolatedTransformation() [2/2]

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

6.1.1.11 optixGetInterpolatedTransformationFromHandle()

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

6.1.1.12 optixGetMatrixFromSrt()

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

6.1.1.13 optixGetObjectToWorldTransformMatrix()

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

6.1.1.14 optixGetWorldToObjectTransformMatrix()

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

6.1.1.15 optixInvertMatrix()

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

6.1.1.16 optixLdg()

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

6.1.1.17 optixLoadInterpolatedMatrixKey()

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

6.1.1.18 optixLoadInterpolatedSrtKey()

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

6.1.1.19 optixLoadReadOnlyAlign16()

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

6.1.1.20 optixMulFloat4()

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

6.1.1.21 optixMultiplyRowMatrix()

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

6.1.1.22 optixResolveMotionKey()

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

6.1.1.23 optixTransformNormal()

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

6.1.1.24 optixTransformPoint()

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

6.1.1.25 optixTransformVector()

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

6.2 optix_internal Namespace Reference

Classes

- struct TypePack

7 Class Documentation

7.1 OptixAabb Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- float minX
- float minY
- float minZ
- float maxX
- float maxY
- float maxZ

7.1.1 Detailed Description

AABB inputs.

7.2 OptixAccelBufferSizes Struct Reference

`#include <optix_7_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( )

7.3 OptixAccelBuildOptions Struct Reference

`#include <optix_7_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( )

7.4 OptixAccelEmitDesc Struct Reference

`#include <optix_7_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( )

7.5 OptixBuildInput Struct Reference

`#include <optix_7_types.h>`

Public Attributes

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

7.5.1 Detailed Description

Build inputs.

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

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( )

7.6 OptixBuildInputCurveArray Struct Reference

`#include <optix_7_types.h>`

Public Attributes

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

7.6.1 Detailed Description

Curve inputs.

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

OptiX describes the curve array as a list of curve segments. The primitive id is the segment number. It is the user's responsibility to maintain a mapping between curves and curve segments. Each index buffer entry i = indexBuffer[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

7.7 OptixBuildInputCustomPrimitiveArray Struct Reference

`#include <optix_7_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

7.8 OptixBuildInputInstanceArray Struct Reference

`#include <optix_7_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

7.9 OptixBuildInputOpacityMicromap Struct Reference

`#include <optix_7_types.h>`

Public Attributes

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

7.10 OptixBuildInputSphereArray Struct Reference

`#include <optix_7_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

7.11 OptixBuildInputTriangleArray Struct Reference

`#include <optix_7_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

7.12 OptixBuiltinISOptions Struct Reference

`#include <optix_7_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( )

7.13 OptixDenoiserGuideLayer Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixImage2D albedo
- OptixImage2D normal
- OptixImage2D flow
- OptixImage2D previousOutputInternalGuideLayer
- OptixImage2D outputInternalGuideLayer

7.13.1 Detailed Description

Guide layer for the denoiser.

See also optixDenoiserInvoke( )

7.14 OptixDenoiserLayer Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixImage2D input
- OptixImage2D previousOutput
- OptixImage2D output

7.14.1 Detailed Description

Input/Output layers for the denoiser.

See also optixDenoiserInvoke( )

7.15 OptixDenoiserOptions Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- unsigned int guideAlbedo
- unsigned int guideNormal

7.15.1 Detailed Description

Options used by the denoiser.

See also optixDenoiserCreate( )

7.16 OptixDenoiserParams Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixDenoiserAlphaMode denoiseAlpha
- CUdeviceptr hdrIntensity
- float blendFactor
- CUdeviceptr hdrAverageColor
- unsigned int temporalModeUsePreviousLayers

7.17 OptixDenoiserSizes Struct Reference

`#include <optix_7_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

7.17.1 Detailed Description

Various sizes related to the denoiser.

See also optixDenoiserComputeMemoryResources( )

7.18 OptixDeviceContextOptions Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixLogCallback logCallbackFunction
- void * logCallbackData
- int logCallbackLevel
- OptixDeviceContextValidationMode validationMode

7.18.1 Detailed Description

Parameters used for optixDeviceContextCreate( )

See also optixDeviceContextCreate( )

7.19 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(∗ optixModuleCreateFromPTX)(OptixDeviceContext context, const OptixModuleCompileOptions ∗moduleCompileOptions, const OptixPipelineCompileOptions ∗pipelineCompileOptions, const char ∗PTX, size_t PTXsize, char ∗logString, size_t ∗logStringSize, OptixModule ∗module)
- OptixResult(∗ optixModuleCreateFromPTXWithTasks)(OptixDeviceContext context, const OptixModuleCompileOptions ∗moduleCompileOptions, const OptixPipelineCompileOptions ∗pipelineCompileOptions, const char ∗PTX, size_t PTXsize, char ∗logString, size_t ∗logStringSize, OptixModule ∗module, OptixTask ∗firstTask)
- OptixResult(∗ optixModuleGetCompilationState)(OptixModule module, OptixModuleCompileState ∗state)
- OptixResult(∗ optixModuleDestroy)(OptixModule module)
- OptixResult(∗ optixBuiltinISModuleGet)(OptixDeviceContext context, const OptixModuleCompileOptions ∗moduleCompileOptions, const OptixPipelineCompileOptions ∗pipelineCompileOptions, const OptixBuiltinISOptions ∗builtinISOptions, OptixModule ∗builtinModule)

Tasks

- OptixResult(∗ optixTaskExecute)(OptixTask task, OptixTask ∗additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int ∗numAdditionalTasksCreated)

Program groups

- OptixResult(∗ optixProgramGroupCreate)(OptixDeviceContext context, const OptixProgramGroupDesc ∗programDescriptions, unsigned int numProgramGroups, const OptixProgramGroupOptions ∗options, char ∗logString, size_t ∗logStringSize, OptixProgramGroup ∗programGroups)
- OptixResult(∗ optixProgramGroupDestroy)(OptixProgramGroup programGroup)
- OptixResult(∗ optixProgramGroupGetStackSize)(OptixProgramGroup programGroup, OptixStackSizes ∗stackSizes)

Pipeline

- OptixResult(∗ optixPipelineCreate)(OptixDeviceContext context, const OptixPipelineCompileOptions ∗pipelineCompileOptions, const OptixPipelineLinkOptions ∗pipelineLinkOptions, const OptixProgramGroup ∗programGroups, unsigned int numProgramGroups, char ∗logString, size_t ∗logStringSize, OptixPipeline ∗pipeline)

- OptixResult(∗ optixPipelineDestroy)(OptixPipeline pipeline)
- OptixResult(∗ optixPipelineSetStackSize)(OptixPipeline pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)

Acceleration structures

- OptixResult(∗ optixAccelComputeMemoryUsage)(OptixDeviceContext context, const OptixAccelBuildOptions ∗accelOptions, const OptixBuildInput ∗buildInputs, unsigned int numBuildInputs, OptixAccelBufferSizes ∗bufferSizes)
- OptixResult(∗ optixAccelBuild)(OptixDeviceContext context, CUstream stream, const OptixAccelBuildOptions ∗accelOptions, const OptixBuildInput ∗buildInputs, unsigned int numBuildInputs, CUdeviceptr tempBuffer, size_t tempBufferSizeInBytes, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle ∗outputHandle, const OptixAccelEmitDesc ∗emittedProperties, unsigned int numEmittedProperties)
- OptixResult(∗ optixAccelGetRelocationInfo)(OptixDeviceContext context, OptixTraversableHandle handle, OptixRelocationInfo ∗info)
- OptixResult(∗ optixCheckRelocationCompatibility)(OptixDeviceContext context, const OptixRelocationInfo ∗info, int ∗compatible)
- OptixResult(∗ optixAccelRelocate)(OptixDeviceContext context, CUstream stream, const OptixRelocationInfo ∗info, const OptixRelocateInput ∗relocateInputs, size_t numRelocateInputs, CUdeviceptr targetAccel, size_t targetAccelSizeInBytes, OptixTraversableHandle ∗targetHandle)
- OptixResult(∗ optixAccelCompact)(OptixDeviceContext context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle ∗outputHandle)
- OptixResult(∗ 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)
- void(∗ reserved1)(void)
- void(∗ reserved2)(void)

Launch

- OptixResult(∗ optixSbtRecordPackHeader)(OptixProgramGroup programGroup, void ∗sbtRecordHeaderHostPointer)
- OptixResult(∗ optixLaunch)(OptixPipeline pipeline, CUstream stream, CUdeviceptr pipelineParams, size_t pipelineParamsSize, const OptixShaderBindingTable ∗sbt, unsigned int width, unsigned int height, unsigned int depth)

Denoiser

- OptixResult(∗ optixDenoiserCreate)(OptixDeviceContext context, OptixDenoiserModelKind modelKind, const OptixDenoiserOptions ∗options, OptixDenoiser ∗returnHandle)
- OptixResult(∗ optixDenoiserDestroy)(OptixDenoiser handle)

- OptixResult(∗ optixDenoiserComputeMemoryResources)(const OptixDenoiser handle, unsigned int maximumInputWidth, unsigned int maximumInputHeight, OptixDenoiserSizes ∗returnSizes)
- OptixResult(∗ optixDenoiserSetup)(OptixDenoiser denoiser, CUstream stream, unsigned int inputWidth, unsigned int inputHeight, CUdeviceptr state, size_t stateSizeInBytes, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OptixResult(∗ optixDenoiserInvoke)(OptixDenoiser denoiser, CUstream stream, const OptixDenoiserParams ∗params, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, const OptixDenoiserGuideLayer ∗guideLayer, const OptixDenoiserLayer ∗layers, unsigned int numLayers, unsigned int inputOffsetX, unsigned int inputOffsetY, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OptixResult(∗ optixDenoiserComputeIntensity)(OptixDenoiser handle, CUstream stream, const OptixImage2D ∗inputImage, CUdeviceptr outputIntensity, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OptixResult(∗ optixDenoiserComputeAverageColor)(OptixDenoiser handle, CUstream stream, const OptixImage2D ∗inputImage, CUdeviceptr outputAverageColor, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OptixResult(∗ optixDenoiserCreateWithUserModel)(OptixDeviceContext context, const void ∗data, size_t dataSizeInBytes, OptixDenoiser ∗returnHandle)

7.19.1 Detailed Description

The function table containing all API functions.

See optixInit( ) and optixInitWithHandle( ).

7.20 OptixImage2D Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- CUdeviceptr data
- unsigned int width
- unsigned int height
- unsigned int rowStrideInBytes
- unsigned int pixelStrideInBytes
- OptixPixelFormat format

7.20.1 Detailed Description

Image descriptor used by the denoiser.

See also optixDenoiserInvoke( ), optixDenoiserComputeIntensity( )

7.21 OptixInstance Struct Reference

`#include <optix_7_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]

7.21.1 Detailed Description

Instances.

See also OptixBuildInputInstanceArray::instances

7.22 OptixMatrixMotionTransform Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixTraversableHandle child
- OptixMotionOptions motionOptions
- unsigned int pad [3]
- float transform [2][12]

7.22.1 Detailed Description

Represents a matrix motion transformation.

The device address of instances of this type must be a multiple of OPTIX_TRANSFORM_BYTE_ALIGNMENT.

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

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

See also optixConvertPointerToTraversableHandle( )

7.23 OptixMicromapBuffers Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- CUdeviceptr output
- size_t outputSizeInBytes
- CUdeviceptr temp
- size_t tempSizeInBytes

7.23.1 Detailed Description

Buffer inputs for opacity micromap array builds.

7.24 OptixMicromapBufferSizes Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- size_t outputSizeInBytes
- size_t tempSizeInBytes

7.24.1 Detailed Description

Conservative memory requirements for building a opacity micromap array.

7.25 OptixModuleCompileBoundValueEntry Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- size_t pipelineParamOffsetInBytes
- size_t sizeInBytes
- const void * boundValuePtr
- const char * annotation

7.25.1 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 consants 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 optixModuleCreateFromPTX 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 optixModuleCreateFromPTX.

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

7.26 OptixModuleCompileOptions Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- int maxRegisterCount
- OptixCompileOptimizationLevel optLevel

- OptixCompileDebugLevel debugLevel
- const OptixModuleCompileBoundValueEntry * boundValues
- unsigned int numBoundValues
- unsigned int numPayloadTypes
- OptixPayloadType * payloadTypes

7.26.1 Detailed Description

Compilation options for module.

See also optixModuleCreateFromPTX( )

7.27 OptixMotionOptions Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- unsigned short numKeys
- unsigned short flags
- float timeBegin
- float timeEnd

7.27.1 Detailed Description

Motion options.

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

7.28 OptixOpacityMicromapArrayBuildInput Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixOpacityMicromapFlags flags
- CUdeviceptr inputBuffer
- CUdeviceptr perMicromapDescBuffer
- unsigned int perMicromapDescStrideInBytes
- unsigned int numMicromapHistogramEntries
- const OptixOpacityMicromapHistogramEntry * micromapHistogramEntries

7.28.1 Detailed Description

Inputs to opacity micromap array construction.

7.29 OptixOpacityMicromapDesc Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- unsigned int byteOffset
- unsigned short subdivisionLevel
- unsigned short format

7.29.1 Detailed Description

Opacity micromap descriptor.

7.30 OptixOpacityMicromapHistogramEntry Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- unsigned int count
- unsigned int subdivisionLevel
- OptixOpacityMicromapFormat format

7.30.1 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.

7.31 OptixOpacityMicromapUsageCount Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- unsigned int count
- unsigned int subdivisionLevel
- OptixOpacityMicromapFormat format

7.31.1 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.

7.32 OptixPayloadType Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- unsigned int numPayloadValues
- const unsigned int * payloadSemantics

7.32.1 Detailed Description

Specifies a single payload type.

7.33 OptixPipelineCompileOptions Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- int usesMotionBlur

- unsigned int traversableGraphFlags
- int numPayloadValues
- int numAttributeValues
- unsigned int exceptionFlags
- const char * pipelineLaunchParamsVariableName
- unsigned int usesPrimitiveTypeFlags
- int allowOpacityMicromaps

7.33.1 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 optixModuleCreateFromPTX( ), optixPipelineCreate( )

7.34 OptixPipelineLinkOptions Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- unsigned int maxTraceDepth
- OptixCompileDebugLevel debugLevel

7.34.1 Detailed Description

Link options for a pipeline.

See also optixPipelineCreate( )

7.35 OptixProgramGroupCallables Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixModule moduleDC
- const char * entryFunctionNameDC
- OptixModule moduleCC
- const char * entryFunctionNameCC

7.35.1 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

7.36 OptixProgramGroupDesc Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixProgramGroupKind kind

- unsigned int flags
- union {
  OptixProgramGroupSingleModule OptixProgramGroupDesc::raygen
  OptixProgramGroupSingleModule OptixProgramGroupDesc::miss
  OptixProgramGroupSingleModule OptixProgramGroupDesc::exception
  OptixProgramGroupCallables OptixProgramGroupDesc::callables
  OptixProgramGroupHitgroup OptixProgramGroupDesc::hitgroup
  };

7.36.1 Detailed Description

Descriptor for program groups.

7.37 OptixProgramGroupHitgroup Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixModule moduleCH
- const char * entryFunctionNameCH
- OptixModule moduleAH
- const char * entryFunctionNameAH
- OptixModule moduleIS
- const char * entryFunctionNameIS

7.37.1 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

7.38 OptixProgramGroupOptions Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixPayloadType * payloadType

7.38.1 Detailed Description

Program group options.

See also optixProgramGroupCreate( )

7.39 OptixProgramGroupSingleModule Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixModule module
- const char * entryFunctionName

7.39.1 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

7.40 OptixRelocateInput Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixBuildInputType type
- union {
  OptixRelocateInputInstanceArray OptixRelocateInput::instanceArray
  OptixRelocateInputTriangleArray OptixRelocateInput::triangleArray
  };

7.40.1 Detailed Description

Relocation inputs.

See also optixAccelRelocate( )

7.41 OptixRelocateInputInstanceArray Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- unsigned int numInstances
- CUdeviceptr traversableHandles

7.41.1 Detailed Description

Instance and instance pointer inputs.

See also OptixRelocateInput::instanceArray

7.42 OptixRelocateInputOpacityMicromap Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- CUdeviceptr opacityMicromapArray

7.43 OptixRelocateInputTriangleArray Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- unsigned int numSbtRecords

- OptixRelocateInputOpacityMicromap opacityMicromap

7.43.1 Detailed Description

Triangle inputs.

See also OptixRelocateInput::triangleArray

7.44 OptixRelocationInfo Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- unsigned long long info [4]

7.44.1 Detailed Description

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

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

7.45 OptixShaderBindingTable Struct Reference

`#include <optix_7_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

7.45.1 Detailed Description

Describes the shader binding table (SBT)

See also optixLaunch( )

7.46 OptixSRTData Struct Reference

`#include <optix_7_types.h>`

Public Attributes

Parameters describing the SRT transformation

- float sx
- float a
- float b
- float pvx
- float sy
- float c
- float pvy
- float sz
- float pvz
- float qx
- float qy
- float qz
- float qw
- float tx
- float ty
- float tz

7.46.1 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( )

7.47 OptixSRTMotionTransform Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixTraversableHandle child
- OptixMotionOptions motionOptions

- unsigned int pad [3]
- OptixSRTData srtData [2]

7.47.1 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( )

7.48 OptixStackSizes Struct Reference

`#include <optix_7_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

7.48.1 Detailed Description

Describes the stack size requirements of a program group.

See also optixProgramGroupGetStackSize( )

7.49 OptixStaticTransform Struct Reference

`#include <optix_7_types.h>`

Public Attributes

- OptixTraversableHandle child
- unsigned int pad [2]
- float transform [12]
- float invTransform [12]

7.49.1 Detailed Description

Static transform.

The device address of instances of this type must be a multiple of OPTIX_TRANSFORM_BYTE_ALIGNMENT.

See also optixConvertPointerToTraversableHandle( )

7.50 OptixUtilDenoiserImageTile Struct Reference

`#include <optix_denoiser_tiling.h>`

Public Attributes

- OptixImage2D input
- OptixImage2D output
- unsigned int inputOffsetX
- unsigned int inputOffsetY

7.50.1 Detailed Description

Tile definition.

see optixUtilDenoiserSplitImage

7.51 optix_internal::TypePack<... > Struct Template Reference

`#include <optix_7_device_impl.h>`

8 File Documentation

8.1 optix_7_device_impl.h File Reference

Classes

- struct optix_internal::TypePack<... >

Namespaces

- namespace optix_internal

Macros

- #define OPTIX_DEFINE_optixGetAttribute_BODY(which)
- #define OPTIX_DEFINE_optixGetExceptionDetail_BODY(which)

Functions

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

- static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_5 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_6 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_7 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_8 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_9 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_10 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_11 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_12 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_13 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_14 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_15 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_16 (unsigned int p)
- 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 ()
- 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 optixGetLinearCurveVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetCubicBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetSphereData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle ()
- static __forceinline__ __device__ float optixGetGASMotionTimeBegin (OptixTraversableHandle handle)
- static __forceinline__ __device__ float optixGetGASMotionTimeEnd (OptixTraversableHandle handle)
- static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (float m[12])
- static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (float3 point)

- static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (float3 normal)
- static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (float3 normal)
- static __forceinline__ __device__ unsigned int optixGetTransformListSize ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTransformType optixGetTransformTypeFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixStaticTransform * optixGetStaticTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixSRTMotionTransform * optixGetSRTMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixMatrixMotionTransform * optixGetMatrixMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceChildFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceInverseTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6, unsigned int a7)
- static __forceinline__ __device__ unsigned int optixGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_1 ()

- static __forceinline__ __device__ unsigned int optixGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_7 ()
- static __forceinline__ __device__ void optixTerminateRay ()
- static __forceinline__ __device__ void optixIgnoreIntersection ()
- static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixGetHitKind ()
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsBackFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsFrontFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ()
- static __forceinline__ __device__ bool optixIsBackFaceHit ()
- static __forceinline__ __device__ bool optixIsFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleHit ()
- static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ()
- static __forceinline__ __device__ float optixGetCurveParameter ()
- static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ()
- static __forceinline__ __device__ uint3 optixGetLaunchIndex ()
- static __forceinline__ __device__ uint3 optixGetLaunchDimensions ()
- static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ()
- static __forceinline__ __device__ void optixThrowException (int exceptionCode)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6, unsigned int exceptionDetail7)

- static __forceinline__ __device__ int optixGetExceptionCode ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7 ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable ()
- static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset ()
- static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay ()
- static __forceinline__ __device__ OptixParameterMismatchExceptionDetails optixGetExceptionParameterMismatch ()
- static __forceinline__ __device__ char * optixGetExceptionLineInfo ()
- template<typename ReturnT , typename... ArgTypes> static __forceinline__ __device__ ReturnT optixDirectCall (unsigned int sbtIndex, ArgTypes... args)
- template<typename ReturnT , typename... ArgTypes> static __forceinline__ __device__ ReturnT optixContinuationCall (unsigned int sbtIndex, ArgTypes... args)
- static __forceinline__ __device__ uint4 optixTexFootprint2D (unsigned long long tex, unsigned int texInfo, float x, float y, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DGrad (unsigned long long tex, unsigned int texInfo, float x, float y, float dPdx_x, float dPdx_y, float dPdy_x, float dPdy_y, bool coarse, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DLod (unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool coarse, unsigned int *singleMipLevel)

8.1.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Device side implementation

8.1.2 Macro Definition Documentation

8.1.2.1 OPTIX_DEFINE_optixGetAttribute_BODY

```
#define OPTIX_DEFINE_optixGetAttribute_BODY(
            which )
```

Value:

```
    unsigned int ret;
\
    asm("call (%0), _optix_get_attribute_" #which ", ();" : "=r"(ret) :);
\
    return ret;
```

8.1.2.2 OPTIX_DEFINE_optixGetExceptionDetail_BODY

```
#define OPTIX_DEFINE_optixGetExceptionDetail_BODY(
            which )
```

Value:

```
    unsigned int ret;
\
    asm("call (%0), _optix_get_exception_detail_" #which ", ();" : "=r"(ret) :);
\
    return ret;
```

8.1.3 Function Documentation

8.1.3.1 optixContinuationCall()

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

8.1.3.2 optixDirectCall()

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

8.1.3.3 optixGetAttribute_0()

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

8.1.3.4 optixGetAttribute_1()

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

8.1.3.5 optixGetAttribute_2()

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

8.1.3.6 optixGetAttribute_3()

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

8.1.3.7 optixGetAttribute_4()

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

8.1.3.8 optixGetAttribute_5()

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

8.1.3.9 optixGetAttribute_6()

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

8.1.3.10 optixGetAttribute_7()

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

8.1.3.11 optixGetCatmullRomVertexData()

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

8.1.3.12 optixGetCubicBSplineVertexData()

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

8.1.3.13 optixGetCurveParameter()

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

8.1.3.14 optixGetExceptionCode()

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

8.1.3.15 optixGetExceptionDetail_0()

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

8.1.3.16 optixGetExceptionDetail_1()

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

8.1.3.17 optixGetExceptionDetail_2()

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

8.1.3.18 optixGetExceptionDetail_3()

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

8.1.3.19 optixGetExceptionDetail_4()

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

[static]

8.1.3.20 optixGetExceptionDetail_5()

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

8.1.3.21 optixGetExceptionDetail_6()

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

8.1.3.22 optixGetExceptionDetail_7()

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

8.1.3.23 optixGetExceptionInvalidRay()

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

8.1.3.24 optixGetExceptionInvalidSbtOffset()

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

8.1.3.25 optixGetExceptionInvalidTraversable()

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

8.1.3.26 optixGetExceptionLineInfo()

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

8.1.3.27 optixGetExceptionParameterMismatch()

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

8.1.3.28 optixGetGASMotionStepCount()

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

8.1.3.29 optixGetGASMotionTimeBegin()

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

8.1.3.30 optixGetGASMotionTimeEnd()

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

8.1.3.31 optixGetGASTraversableHandle()

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

8.1.3.32 optixGetHitKind()

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

8.1.3.33 optixGetInstanceChildFromHandle()

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

8.1.3.34 optixGetInstanceId()

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

8.1.3.35 optixGetInstanceIdFromHandle()

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

8.1.3.36 optixGetInstanceIndex()

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

8.1.3.37 optixGetInstanceInverseTransformFromHandle()

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

8.1.3.38 optixGetInstanceTransformFromHandle()

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

8.1.3.39 optixGetInstanceTraversableFromIAS()

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

8.1.3.40 optixGetLaunchDimensions()

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

8.1.3.41 optixGetLaunchIndex()

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

8.1.3.42 optixGetLinearCurveVertexData()

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

8.1.3.43 optixGetMatrixMotionTransformFromHandle()

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

8.1.3.44 optixGetObjectRayDirection()

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

8.1.3.45 optixGetObjectRayOrigin()

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

8.1.3.46 optixGetObjectToWorldTransformMatrix()

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

8.1.3.47 optixGetPayload_0()

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

8.1.3.48 optixGetPayload_1()

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

8.1.3.49 optixGetPayload_10()

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

8.1.3.50 optixGetPayload_11()

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

8.1.3.51 optixGetPayload_12()

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

8.1.3.52 optixGetPayload_13()

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

8.1.3.53 optixGetPayload_14()

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

8.1.3.54 optixGetPayload_15()

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

8.1.3.55 optixGetPayload_16()

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

8.1.3.56 optixGetPayload_17()

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

8.1.3.57 optixGetPayload_18()

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

8.1.3.58 optixGetPayload_19()

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

8.1.3.59 optixGetPayload_2()

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

8.1.3.60 optixGetPayload_20()

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

8.1.3.61 optixGetPayload_21()

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

8.1.3.62 optixGetPayload_22()

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

8.1.3.63 optixGetPayload_23()

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

8.1.3.64 optixGetPayload_24()

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

8.1.3.65 optixGetPayload_25()

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

8.1.3.66 optixGetPayload_26()

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

8.1.3.67 optixGetPayload_27()

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

8.1.3.68 optixGetPayload_28()

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

8.1.3.69 optixGetPayload_29()

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

8.1.3.70 optixGetPayload_3()

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

8.1.3.71 optixGetPayload_30()

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

8.1.3.72 optixGetPayload_31()

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

8.1.3.73 optixGetPayload_4()

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

8.1.3.74 optixGetPayload_5()

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

8.1.3.75 optixGetPayload_6()

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

8.1.3.76 optixGetPayload_7()

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

8.1.3.77 optixGetPayload_8()

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

8.1.3.78 optixGetPayload_9()

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

8.1.3.79 optixGetPrimitiveIndex()

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

8.1.3.80 optixGetPrimitiveType() [1/2]

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

8.1.3.81 optixGetPrimitiveType() [2/2]

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

8.1.3.82 optixGetQuadraticBSplineVertexData()

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

8.1.3.83 optixGetRayFlags()

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

8.1.3.84 optixGetRayTime()

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

8.1.3.85 optixGetRayTmax()

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

8.1.3.86 optixGetRayTmin()

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

8.1.3.87 optixGetRayVisibilityMask()

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

8.1.3.88 optixGetSbtDataPointer()

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

8.1.3.89 optixGetSbtGASIndex()

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

8.1.3.90 optixGetSphereData()

```
static __forceinline__ __device__ void optixGetSphereData (
          OptixTraversableHandle gas,
          unsigned int primIdx,
```

```
          unsigned int sbtGASIndex,
          float time,
          float4 data[1] )  [static]
```

8.1.3.91 optixGetSRTMotionTransformFromHandle()

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

8.1.3.92 optixGetStaticTransformFromHandle()

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

8.1.3.93 optixGetTransformListHandle()

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

8.1.3.94 optixGetTransformListSize()

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

8.1.3.95 optixGetTransformTypeFromHandle()

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

8.1.3.96 optixGetTriangleBarycentrics()

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

8.1.3.97 optixGetTriangleVertexData()

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

8.1.3.98 optixGetWorldRayDirection()

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

8.1.3.99 optixGetWorldRayOrigin()

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

8.1.3.100 optixGetWorldToObjectTransformMatrix()

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

8.1.3.101 optixIgnoreIntersection()

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

8.1.3.102 optixIsBackFaceHit() [1/2]

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

8.1.3.103 optixIsBackFaceHit() [2/2]

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

8.1.3.104 optixIsFrontFaceHit() [1/2]

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

8.1.3.105 optixIsFrontFaceHit() [2/2]

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

8.1.3.106 optixIsTriangleBackFaceHit()

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

8.1.3.107 optixIsTriangleFrontFaceHit()

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

8.1.3.108 optixIsTriangleHit()

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

8.1.3.109 optixReportIntersection() [1/9]

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

8.1.3.110 optixReportIntersection() [2/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
          float hitT,
```

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

8.1.3.111 optixReportIntersection() [3/9]

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

8.1.3.112 optixReportIntersection() [4/9]

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

8.1.3.113 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]
```

8.1.3.114 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]
```

8.1.3.115 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]
```

8.1.3.116 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]
```

8.1.3.117 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]
```

8.1.3.118 optixSetPayload_0()

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

8.1.3.119 optixSetPayload_1()

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

8.1.3.120 optixSetPayload_10()

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

8.1.3.121 optixSetPayload_11()

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

8.1.3.122 optixSetPayload_12()

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

8.1.3.123 optixSetPayload_13()

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

8.1.3.124 optixSetPayload_14()

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

8.1.3.125 optixSetPayload_15()

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

8.1.3.126 optixSetPayload_16()

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

8.1.3.127 optixSetPayload_17()

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

8.1.3.128 optixSetPayload_18()

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

8.1.3.129 optixSetPayload_19()

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

8.1.3.130 optixSetPayload_2()

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

8.1.3.131 optixSetPayload_20()

```
static __forceinline__ __device__ void optixSetPayload_20 (
```

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

8.1.3.132 optixSetPayload_21()

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

8.1.3.133 optixSetPayload_22()

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

8.1.3.134 optixSetPayload_23()

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

8.1.3.135 optixSetPayload_24()

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

8.1.3.136 optixSetPayload_25()

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

8.1.3.137 optixSetPayload_26()

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

8.1.3.138 optixSetPayload_27()

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

8.1.3.139 optixSetPayload_28()

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

8.1.3.140 optixSetPayload_29()

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

8.1.3.141 optixSetPayload_3()

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

8.1.3.142 optixSetPayload_30()

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

8.1.3.143 optixSetPayload_31()

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

8.1.3.144 optixSetPayload_4()

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

8.1.3.145 optixSetPayload_5()

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

8.1.3.146 optixSetPayload_6()

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

8.1.3.147 optixSetPayload_7()

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

8.1.3.148 optixSetPayload_8()

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

8.1.3.149 optixSetPayload_9()

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

8.1.3.150 optixSetPayloadTypes()

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

8.1.3.151 optixTerminateRay()

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

8.1.3.152 optixTexFootprint2D()

```
static __forceinline__ __device__ uint4 optixTexFootprint2D (
          unsigned long long tex,
```

```
          unsigned int texInfo,
          float x,
          float y,
          unsigned int * singleMipLevel ) [static]
```

8.1.3.153 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]
```

8.1.3.154 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]
```

8.1.3.155 optixThrowException() [1/9]

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

8.1.3.156 optixThrowException() [2/9]

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

8.1.3.157 optixThrowException() [3/9]

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

8.1.3.158 optixThrowException() [4/9]

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

8.1.3.159 optixThrowException() [5/9]

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

8.1.3.160 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]
```

8.1.3.161 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]
```

8.1.3.162 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]
```

8.1.3.163 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]
```

8.1.3.164 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]
```

8.1.3.165 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]
```

8.1.3.166 optixTransformNormalFromObjectToWorldSpace()

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

8.1.3.167 optixTransformNormalFromWorldToObjectSpace()

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

8.1.3.168 optixTransformPointFromObjectToWorldSpace()

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

8.1.3.169 optixTransformPointFromWorldToObjectSpace()

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

8.1.3.170 optixTransformVectorFromObjectToWorldSpace()

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

8.1.3.171 optixTransformVectorFromWorldToObjectSpace()

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

8.1.3.172 optixUndefinedValue()

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

8.2 optix_7_device_impl.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
3 *
```

```
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly
8 * prohibited.
9 *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
29 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
30 #error("optix_7_device_impl.h is an internal header file and must not be used directly.  Please use
optix_device.h or optix.h instead.")
31 #endif
32
33 #ifndef __optix_optix_7_device_impl_h__
34 #define __optix_optix_7_device_impl_h__
35
36 #include "internal/optix_7_device_impl_exception.h"
37 #include "internal/optix_7_device_impl_transformations.h"
38
39 #ifndef __CUDACC_RTC__
40 #include <initializer_list>
41 #include <type_traits>
42 #endif
43
44 namespace optix_internal {
45 template <typename...>
46 struct TypePack{};
47 }  // namespace optix_internal
48
49 template <typename... Payload>
50 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
51                                                   float3                rayOrigin,
52                                                   float3                rayDirection,
53                                                   float                 tmin,
54                                                   float                 tmax,
55                                                   float                 rayTime,
56                                                   OptixVisibilityMask   visibilityMask,
57                                                   unsigned int          rayFlags,
58                                                   unsigned int          SBToffset,
59                                                   unsigned int          SBTstride,
60                                                   unsigned int          missSBTIndex,
61                                                   Payload&...             payload)
62 {
63     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
64     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
65     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
66     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
67 #ifndef __CUDACC_RTC__
68     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int»::value,
69                   "All payload parameters need to be unsigned int.");
70 #endif
71
72     float        ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
73     float        dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
74     unsigned int p[33]       = { 0, payload... };
75     int          payloadSize = (int)sizeof...(Payload);
76     asm volatile(
```

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

```
139         : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
140           "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBToffset), "r"(SBTstride),
141           "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
142           "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
143           "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
144           "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
145           "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
146         :);
147     unsigned int index = 1;
148     (void)std::initializer_list<unsigned int>{ index, (payload = p[index++])... };
149 }
150
151 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p)
152 {
153     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(0), "r"(p) :);
154 }
155
156 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p)
157 {
158     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(1), "r"(p) :);
159 }
160
161 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p)
162 {
163     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(2), "r"(p) :);
164 }
165
166 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p)
167 {
168     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(3), "r"(p) :);
169 }
170
171 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p)
172 {
173     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(4), "r"(p) :);
174 }
175
176 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p)
177 {
178     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(5), "r"(p) :);
179 }
180
181 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p)
182 {
183     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(6), "r"(p) :);
184 }
185
186 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p)
187 {
188     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(7), "r"(p) :);
189 }
190
191 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p)
192 {
193     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(8), "r"(p) :);
194 }
195
196 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p)
197 {
198     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(9), "r"(p) :);
199 }
200
201 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p)
202 {
203     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(10), "r"(p) :);
204 }
205
```

```
206 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p)
207 {
208     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(11), "r"(p) :);
209 }
210
211 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p)
212 {
213     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(12), "r"(p) :);
214 }
215
216 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p)
217 {
218     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(13), "r"(p) :);
219 }
220
221 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p)
222 {
223     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(14), "r"(p) :);
224 }
225
226 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p)
227 {
228     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(15), "r"(p) :);
229 }
230
231 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p)
232 {
233     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(16), "r"(p) :);
234 }
235
236 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p)
237 {
238     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(17), "r"(p) :);
239 }
240
241 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p)
242 {
243     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(18), "r"(p) :);
244 }
245
246 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p)
247 {
248     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(19), "r"(p) :);
249 }
250
251 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p)
252 {
253     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(20), "r"(p) :);
254 }
255
256 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p)
257 {
258     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(21), "r"(p) :);
259 }
260
261 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p)
262 {
263     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(22), "r"(p) :);
264 }
265
266 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p)
267 {
268     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(23), "r"(p) :);
269 }
270
271 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p)
272 {
```

```
273     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(24), "r"(p) :);
274 }
275
276 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p)
277 {
278     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(25), "r"(p) :);
279 }
280
281 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p)
282 {
283     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(26), "r"(p) :);
284 }
285
286 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p)
287 {
288     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(27), "r"(p) :);
289 }
290
291 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p)
292 {
293     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(28), "r"(p) :);
294 }
295
296 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p)
297 {
298     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(29), "r"(p) :);
299 }
300
301 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p)
302 {
303     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(30), "r"(p) :);
304 }
305
306 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p)
307 {
308     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(31), "r"(p) :);
309 }
310
311 static __forceinline__ __device__ unsigned int optixGetPayload_0()
312 {
313     unsigned int result;
314     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(0) :);
315     return result;
316 }
317
318 static __forceinline__ __device__ unsigned int optixGetPayload_1()
319 {
320     unsigned int result;
321     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(1) :);
322     return result;
323 }
324
325 static __forceinline__ __device__ unsigned int optixGetPayload_2()
326 {
327     unsigned int result;
328     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(2) :);
329     return result;
330 }
331
332 static __forceinline__ __device__ unsigned int optixGetPayload_3()
333 {
334     unsigned int result;
335     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(3) :);
336     return result;
337 }
338
339 static __forceinline__ __device__ unsigned int optixGetPayload_4()
```

```
340 {
341     unsigned int result;
342     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(4) :);
343     return result;
344 }
345
346 static __forceinline__ __device__ unsigned int optixGetPayload_5()
347 {
348     unsigned int result;
349     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(5) :);
350     return result;
351 }
352
353 static __forceinline__ __device__ unsigned int optixGetPayload_6()
354 {
355     unsigned int result;
356     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(6) :);
357     return result;
358 }
359
360 static __forceinline__ __device__ unsigned int optixGetPayload_7()
361 {
362     unsigned int result;
363     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(7) :);
364     return result;
365 }
366
367 static __forceinline__ __device__ unsigned int optixGetPayload_8()
368 {
369     unsigned int result;
370     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(8) :);
371     return result;
372 }
373
374 static __forceinline__ __device__ unsigned int optixGetPayload_9()
375 {
376     unsigned int result;
377     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(9) :);
378     return result;
379 }
380
381 static __forceinline__ __device__ unsigned int optixGetPayload_10()
382 {
383     unsigned int result;
384     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(10) :);
385     return result;
386 }
387
388 static __forceinline__ __device__ unsigned int optixGetPayload_11()
389 {
390     unsigned int result;
391     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(11) :);
392     return result;
393 }
394
395 static __forceinline__ __device__ unsigned int optixGetPayload_12()
396 {
397     unsigned int result;
398     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(12) :);
399     return result;
400 }
401
402 static __forceinline__ __device__ unsigned int optixGetPayload_13()
403 {
404     unsigned int result;
405     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(13) :);
406     return result;
```

```
407 }
408
409 static __forceinline__ __device__ unsigned int optixGetPayload_14()
410 {
411     unsigned int result;
412     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(14) :);
413     return result;
414 }
415
416 static __forceinline__ __device__ unsigned int optixGetPayload_15()
417 {
418     unsigned int result;
419     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(15) :);
420     return result;
421 }
422
423 static __forceinline__ __device__ unsigned int optixGetPayload_16()
424 {
425     unsigned int result;
426     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(16) :);
427     return result;
428 }
429
430 static __forceinline__ __device__ unsigned int optixGetPayload_17()
431 {
432     unsigned int result;
433     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(17) :);
434     return result;
435 }
436
437 static __forceinline__ __device__ unsigned int optixGetPayload_18()
438 {
439     unsigned int result;
440     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(18) :);
441     return result;
442 }
443
444 static __forceinline__ __device__ unsigned int optixGetPayload_19()
445 {
446     unsigned int result;
447     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(19) :);
448     return result;
449 }
450
451 static __forceinline__ __device__ unsigned int optixGetPayload_20()
452 {
453     unsigned int result;
454     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(20) :);
455     return result;
456 }
457
458 static __forceinline__ __device__ unsigned int optixGetPayload_21()
459 {
460     unsigned int result;
461     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(21) :);
462     return result;
463 }
464
465 static __forceinline__ __device__ unsigned int optixGetPayload_22()
466 {
467     unsigned int result;
468     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(22) :);
469     return result;
470 }
471
472 static __forceinline__ __device__ unsigned int optixGetPayload_23()
473 {
```

```
474     unsigned int result;
475     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(23) :);
476     return result;
477 }
478
479 static __forceinline__ __device__ unsigned int optixGetPayload_24()
480 {
481     unsigned int result;
482     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(24) :);
483     return result;
484 }
485
486 static __forceinline__ __device__ unsigned int optixGetPayload_25()
487 {
488     unsigned int result;
489     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(25) :);
490     return result;
491 }
492
493 static __forceinline__ __device__ unsigned int optixGetPayload_26()
494 {
495     unsigned int result;
496     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(26) :);
497     return result;
498 }
499
500 static __forceinline__ __device__ unsigned int optixGetPayload_27()
501 {
502     unsigned int result;
503     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(27) :);
504     return result;
505 }
506
507 static __forceinline__ __device__ unsigned int optixGetPayload_28()
508 {
509     unsigned int result;
510     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(28) :);
511     return result;
512 }
513
514 static __forceinline__ __device__ unsigned int optixGetPayload_29()
515 {
516     unsigned int result;
517     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(29) :);
518     return result;
519 }
520
521 static __forceinline__ __device__ unsigned int optixGetPayload_30()
522 {
523     unsigned int result;
524     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(30) :);
525     return result;
526 }
527
528 static __forceinline__ __device__ unsigned int optixGetPayload_31()
529 {
530     unsigned int result;
531     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(31) :);
532     return result;
533 }
534
535 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int types)
536 {
537     asm volatile("call _optix_set_payload_types, (%0);" : : "r"(types) :);
538 }
539
540 static __forceinline__ __device__ unsigned int optixUndefinedValue()
```

```
541 {
542     unsigned int u0;
543     asm("call (%0), _optix_undef_value, ();" : "=r"(u0) :);
544     return u0;
545 }
546
547 static __forceinline__ __device__ float3 optixGetWorldRayOrigin()
548 {
549     float f0, f1, f2;
550     asm("call (%0), _optix_get_world_ray_origin_x, ();" : "=f"(f0) :);
551     asm("call (%0), _optix_get_world_ray_origin_y, ();" : "=f"(f1) :);
552     asm("call (%0), _optix_get_world_ray_origin_z, ();" : "=f"(f2) :);
553     return make_float3(f0, f1, f2);
554 }
555
556 static __forceinline__ __device__ float3 optixGetWorldRayDirection()
557 {
558     float f0, f1, f2;
559     asm("call (%0), _optix_get_world_ray_direction_x, ();" : "=f"(f0) :);
560     asm("call (%0), _optix_get_world_ray_direction_y, ();" : "=f"(f1) :);
561     asm("call (%0), _optix_get_world_ray_direction_z, ();" : "=f"(f2) :);
562     return make_float3(f0, f1, f2);
563 }
564
565 static __forceinline__ __device__ float3 optixGetObjectRayOrigin()
566 {
567     float f0, f1, f2;
568     asm("call (%0), _optix_get_object_ray_origin_x, ();" : "=f"(f0) :);
569     asm("call (%0), _optix_get_object_ray_origin_y, ();" : "=f"(f1) :);
570     asm("call (%0), _optix_get_object_ray_origin_z, ();" : "=f"(f2) :);
571     return make_float3(f0, f1, f2);
572 }
573
574 static __forceinline__ __device__ float3 optixGetObjectRayDirection()
575 {
576     float f0, f1, f2;
577     asm("call (%0), _optix_get_object_ray_direction_x, ();" : "=f"(f0) :);
578     asm("call (%0), _optix_get_object_ray_direction_y, ();" : "=f"(f1) :);
579     asm("call (%0), _optix_get_object_ray_direction_z, ();" : "=f"(f2) :);
580     return make_float3(f0, f1, f2);
581 }
582
583 static __forceinline__ __device__ float optixGetRayTmin()
584 {
585     float f0;
586     asm("call (%0), _optix_get_ray_tmin, ();" : "=f"(f0) :);
587     return f0;
588 }
589
590 static __forceinline__ __device__ float optixGetRayTmax()
591 {
592     float f0;
593     asm("call (%0), _optix_get_ray_tmax, ();" : "=f"(f0) :);
594     return f0;
595 }
596
597 static __forceinline__ __device__ float optixGetRayTime()
598 {
599     float f0;
600     asm("call (%0), _optix_get_ray_time, ();" : "=f"(f0) :);
601     return f0;
602 }
603
604 static __forceinline__ __device__ unsigned int optixGetRayFlags()
605 {
606     unsigned int u0;
607     asm("call (%0), _optix_get_ray_flags, ();" : "=r"(u0) :);
```

```
608     return u0;
609 }
610
611 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask( )
612 {
613     unsigned int u0;
614     asm("call (%0), _optix_get_ray_visibility_mask, ();" : "=r"(u0) :);
615     return u0;
616 }
617
618 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias,
619                                                                                 unsigned int
instIdx)
620 {
621     unsigned long long handle;
622     asm("call (%0), _optix_get_instance_traversable_from_ias, (%1, %2);"
623         : "=l"(handle) : "l"(ias), "r"(instIdx));
624     return (OptixTraversableHandle)handle;
625 }
626
627
628 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas,
629                                                                   unsigned int           primIdx,
630                                                                   unsigned int           sbtGASIndex,
631                                                                   float                  time,
632                                                                   float3                 data[3])
633 {
634     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_triangle_vertex_data, "
635         "(%9, %10, %11, %12);"
636         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
637           "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
638         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
639         :);
640 }
641
642
643 static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
644                                                                      unsigned int           primIdx,
645                                                                     unsigned int           sbtGASIndex,
646                                                                      float                  time,
647                                                                      float4                 data[2])
648 {
649     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7), _optix_get_linear_curve_vertex_data, "
650         "(%8, %9, %10, %11);"
651         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
652           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
653         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
654         :);
655 }
656
657 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(OptixTraversableHandle gas,
658                                                                           unsigned int         primIdx,
659                                                                         unsigned int         sbtGASIndex,
660                                                                           float                time,
661                                                                           float4               data[3])
662 {
663     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11),
_optix_get_quadratic_bspline_vertex_data, "
664         "(%12, %13, %14, %15);"
665         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
666           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
667           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w)
668         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
669         :);
670 }
671
```

```
672 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,
673                                                                         unsigned int          primIdx,
674                                                                         unsigned int          sbtGASIndex,
675                                                                         float                 time,
676                                                                         float4                data[4])
677 {
678     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
679         "_optix_get_cubic_bspline_vertex_data, "
680         "(%16, %17, %18, %19);"
681         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
682           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
683           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
684           "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
685         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
686         :);
687 }
688
689 static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas,
690                                                                      unsigned int          primIdx,
691                                                                      unsigned int          sbtGASIndex,
692                                                                      float                 time,
693                                                                      float4                data[4])
694 {
695     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
696         "_optix_get_catmullrom_vertex_data, "
697         "(%16, %17, %18, %19);"
698         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
699           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
700           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
701         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
702         :);
703 }
704
705 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas,
706                                                           unsigned int          primIdx,
707                                                           unsigned int          sbtGASIndex,
708                                                           float                 time,
709                                                           float4                data[1])
710 {
711     asm("call (%0, %1, %2, %3), "
712         "_optix_get_sphere_data, "
713         "(%4, %5, %6, %7);"
714         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
715         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
716         :);
717 }
718
719 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle()
720 {
721     unsigned long long handle;
722     asm("call (%0), _optix_get_gas_traversable_handle, ();" : "=l"(handle) :);
723     return (OptixTraversableHandle)handle;
724 }
725
726 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle handle)
727 {
728     float f0;
729     asm("call (%0), _optix_get_gas_motion_time_begin, (%1);" : "=f"(f0) : "l"(handle) :);
730     return f0;
731 }
732
733 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle handle)
734 {
735     float f0;
736     asm("call (%0), _optix_get_gas_motion_time_end, (%1);" : "=f"(f0) : "l"(handle) :);
737     return f0;
```

```
738 }
739
740 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle handle)
741 {
742     unsigned int u0;
743     asm("call (%0), _optix_get_gas_motion_step_count, (%1);" : "=r"(u0) : "l"(handle) :);
744     return u0;
745 }
746
747 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12])
748 {
749     if(optixGetTransformListSize() == 0)
750     {
751         m[0]  = 1.0f;
752         m[1]  = 0.0f;
753         m[2]  = 0.0f;
754         m[3]  = 0.0f;
755         m[4]  = 0.0f;
756         m[5]  = 1.0f;
757         m[6]  = 0.0f;
758         m[7]  = 0.0f;
759         m[8]  = 0.0f;
760         m[9]  = 0.0f;
761         m[10] = 1.0f;
762         m[11] = 0.0f;
763         return;
764     }
765
766     float4 m0, m1, m2;
767     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
768     m[0]  = m0.x;
769     m[1]  = m0.y;
770     m[2]  = m0.z;
771     m[3]  = m0.w;
772     m[4]  = m1.x;
773     m[5]  = m1.y;
774     m[6]  = m1.z;
775     m[7]  = m1.w;
776     m[8]  = m2.x;
777     m[9]  = m2.y;
778     m[10] = m2.z;
779     m[11] = m2.w;
780 }
781
782 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12])
783 {
784     if(optixGetTransformListSize() == 0)
785     {
786         m[0]  = 1.0f;
787         m[1]  = 0.0f;
788         m[2]  = 0.0f;
789         m[3]  = 0.0f;
790         m[4]  = 0.0f;
791         m[5]  = 1.0f;
792         m[6]  = 0.0f;
793         m[7]  = 0.0f;
794         m[8]  = 0.0f;
795         m[9]  = 0.0f;
796         m[10] = 1.0f;
797         m[11] = 0.0f;
798         return;
799     }
800
801     float4 m0, m1, m2;
802     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);
803     m[0]  = m0.x;
804     m[1]  = m0.y;
```

```
805     m[2]  = m0.z;
806     m[3]  = m0.w;
807     m[4]  = m1.x;
808     m[5]  = m1.y;
809     m[6]  = m1.z;
810     m[7]  = m1.w;
811     m[8]  = m2.x;
812     m[9]  = m2.y;
813     m[10] = m2.z;
814     m[11] = m2.w;
815 }
816
817 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point)
818 {
819     if(optixGetTransformListSize() == 0)
820         return point;
821
822     float4 m0, m1, m2;
823     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
824     return optix_impl::optixTransformPoint(m0, m1, m2, point);
825 }
826
827 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec)
828 {
829     if(optixGetTransformListSize() == 0)
830         return vec;
831
832     float4 m0, m1, m2;
833     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
834     return optix_impl::optixTransformVector(m0, m1, m2, vec);
835 }
836
837 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal)
838 {
839     if(optixGetTransformListSize() == 0)
840         return normal;
841
842     float4 m0, m1, m2;
843     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);  // inverse of
optixGetWorldToObjectTransformMatrix()
844     return optix_impl::optixTransformNormal(m0, m1, m2, normal);
845 }
846
847 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point)
848 {
849     if(optixGetTransformListSize() == 0)
850         return point;
851
852     float4 m0, m1, m2;
853     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);
854     return optix_impl::optixTransformPoint(m0, m1, m2, point);
855 }
856
857 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec)
858 {
859     if(optixGetTransformListSize() == 0)
860         return vec;
861
862     float4 m0, m1, m2;
863     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);
864     return optix_impl::optixTransformVector(m0, m1, m2, vec);
865 }
866
867 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal)
868 {
869     if(optixGetTransformListSize() == 0)
870         return normal;
```

```
871
872     float4 m0, m1, m2;
873     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);  // inverse of
optixGetObjectToWorldTransformMatrix()
874     return optix_impl::optixTransformNormal(m0, m1, m2, normal);
875 }
876
877 static __forceinline__ __device__ unsigned int optixGetTransformListSize()
878 {
879     unsigned int u0;
880     asm("call (%0), _optix_get_transform_list_size, ();" : "=r"(u0) :);
881     return u0;
882 }
883
884 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int index)
885 {
886     unsigned long long u0;
887     asm("call (%0), _optix_get_transform_list_handle, (%1);" : "=l"(u0) : "r"(index) :);
888     return u0;
889 }
890
891 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle)
892 {
893     int i0;
894     asm("call (%0), _optix_get_transform_type_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
895     return (OptixTransformType)i0;
896 }
897
898 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle)
899 {
900     unsigned long long ptr;
901     asm("call (%0), _optix_get_static_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
902     return (const OptixStaticTransform*)ptr;
903 }
904
905 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle)
906 {
907     unsigned long long ptr;
908     asm("call (%0), _optix_get_srt_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
909     return (const OptixSRTMotionTransform*)ptr;
910 }
911
912 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle)
913 {
914     unsigned long long ptr;
915     asm("call (%0), _optix_get_matrix_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
916     return (const OptixMatrixMotionTransform*)ptr;
917 }
918
919 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle)
920 {
921     int i0;
922     asm("call (%0), _optix_get_instance_id_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
923     return i0;
924 }
925
926 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle(OptixTraversableHandle handle)
927 {
928     unsigned long long i0;
929     asm("call (%0), _optix_get_instance_child_from_handle, (%1);" : "=l"(i0) : "l"(handle) :);
930     return (OptixTraversableHandle)i0;
```

```
931 }
932
933 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle)
934 {
935     unsigned long long ptr;
936     asm("call (%0), _optix_get_instance_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
937     return (const float4*)ptr;
938 }
939
940 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle)
941 {
942     unsigned long long ptr;
943     asm("call (%0), _optix_get_instance_inverse_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle)
:);
944     return (const float4*)ptr;
945 }
946
947 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind)
948 {
949     int ret;
950     asm volatile(
951         "call (%0), _optix_report_intersection_0"
952         ", (%1, %2);"
953         : "=r"(ret)
954         : "f"(hitT), "r"(hitKind)
955         :);
956     return ret;
957 }
958
959 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0)
960 {
961     int ret;
962     asm volatile(
963         "call (%0), _optix_report_intersection_1"
964         ", (%1, %2, %3);"
965         : "=r"(ret)
966         : "f"(hitT), "r"(hitKind), "r"(a0)
967         :);
968     return ret;
969 }
970
971 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1)
972 {
973     int ret;
974     asm volatile(
975         "call (%0), _optix_report_intersection_2"
976         ", (%1, %2, %3, %4);"
977         : "=r"(ret)
978         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1)
979         :);
980     return ret;
981 }
982
983 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1, unsigned int a2)
984 {
985     int ret;
986     asm volatile(
987         "call (%0), _optix_report_intersection_3"
988         ", (%1, %2, %3, %4, %5);"
989         : "=r"(ret)
990         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2)
991         :);
```

```
992     return ret;
993 }
994
995 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
996                                                                 unsigned int hitKind,
997                                                                 unsigned int a0,
998                                                                 unsigned int a1,
999                                                                 unsigned int a2,
1000                                                                unsigned int a3)
1001 {
1002     int ret;
1003     asm volatile(
1004         "call (%0), _optix_report_intersection_4"
1005         ", (%1, %2, %3, %4, %5, %6);"
1006         : "=r"(ret)
1007         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3)
1008         :);
1009     return ret;
1010 }
1011
1012 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1013                                                                unsigned int hitKind,
1014                                                                unsigned int a0,
1015                                                                unsigned int a1,
1016                                                                unsigned int a2,
1017                                                                unsigned int a3,
1018                                                                unsigned int a4)
1019 {
1020     int ret;
1021     asm volatile(
1022         "call (%0), _optix_report_intersection_5"
1023         ", (%1, %2, %3, %4, %5, %6, %7);"
1024         : "=r"(ret)
1025         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4)
1026         :);
1027     return ret;
1028 }
1029
1030 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1031                                                                unsigned int hitKind,
1032                                                                unsigned int a0,
1033                                                                unsigned int a1,
1034                                                                unsigned int a2,
1035                                                                unsigned int a3,
1036                                                                unsigned int a4,
1037                                                                unsigned int a5)
1038 {
1039     int ret;
1040     asm volatile(
1041         "call (%0), _optix_report_intersection_6"
1042         ", (%1, %2, %3, %4, %5, %6, %7, %8);"
1043         : "=r"(ret)
1044         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5)
1045         :);
1046     return ret;
1047 }
1048
1049 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1050                                                                unsigned int hitKind,
1051                                                                unsigned int a0,
1052                                                                unsigned int a1,
1053                                                                unsigned int a2,
1054                                                                unsigned int a3,
1055                                                                unsigned int a4,
1056                                                                unsigned int a5,
1057                                                                unsigned int a6)
1058 {
```

```
1059     int ret;
1060     asm volatile(
1061         "call (%0), _optix_report_intersection_7"
1062         ", (%1, %2, %3, %4, %5, %6, %7, %8, %9);"
1063         : "=r"(ret)
1064         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6)
1065         :);
1066     return ret;
1067 }
1068
1069 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1070                                                                unsigned int hitKind,
1071                                                                unsigned int a0,
1072                                                                unsigned int a1,
1073                                                                unsigned int a2,
1074                                                                unsigned int a3,
1075                                                                unsigned int a4,
1076                                                                unsigned int a5,
1077                                                                unsigned int a6,
1078                                                                unsigned int a7)
1079 {
1080     int ret;
1081     asm volatile(
1082         "call (%0), _optix_report_intersection_8"
1083         ", (%1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
1084         : "=r"(ret)
1085        : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6), "r"(a7)
1086         :);
1087     return ret;
1088 }
1089
1090 #define OPTIX_DEFINE_optixGetAttribute_BODY(which)
\
1091 unsigned int ret;
\
1092 asm("call (%0), _optix_get_attribute_" #which ", ();" : "=r"(ret) :);
\
1093     return ret;
1094
1095 static __forceinline__ __device__ unsigned int optixGetAttribute_0()
1096 {
1097     OPTIX_DEFINE_optixGetAttribute_BODY(0);
1098 }
1099
1100 static __forceinline__ __device__ unsigned int optixGetAttribute_1()
1101 {
1102     OPTIX_DEFINE_optixGetAttribute_BODY(1);
1103 }
1104
1105 static __forceinline__ __device__ unsigned int optixGetAttribute_2()
1106 {
1107     OPTIX_DEFINE_optixGetAttribute_BODY(2);
1108 }
1109
1110 static __forceinline__ __device__ unsigned int optixGetAttribute_3()
1111 {
1112     OPTIX_DEFINE_optixGetAttribute_BODY(3);
1113 }
1114
1115 static __forceinline__ __device__ unsigned int optixGetAttribute_4()
1116 {
1117     OPTIX_DEFINE_optixGetAttribute_BODY(4);
1118 }
1119
1120 static __forceinline__ __device__ unsigned int optixGetAttribute_5()
1121 {
1122     OPTIX_DEFINE_optixGetAttribute_BODY(5);
```

```
1123 }
1124
1125 static __forceinline__ __device__ unsigned int optixGetAttribute_6()
1126 {
1127     OPTIX_DEFINE_optixGetAttribute_BODY(6);
1128 }
1129
1130 static __forceinline__ __device__ unsigned int optixGetAttribute_7()
1131 {
1132     OPTIX_DEFINE_optixGetAttribute_BODY(7);
1133 }
1134
1135 #undef OPTIX_DEFINE_optixGetAttribute_BODY
1136
1137 static __forceinline__ __device__ void optixTerminateRay()
1138 {
1139     asm volatile("call _optix_terminate_ray, ();");
1140 }
1141
1142 static __forceinline__ __device__ void optixIgnoreIntersection()
1143 {
1144     asm volatile("call _optix_ignore_intersection, ();");
1145 }
1146
1147 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex()
1148 {
1149     unsigned int u0;
1150     asm("call (%0), _optix_read_primitive_idx, ();" : "=r"(u0) :);
1151     return u0;
1152 }
1153
1154 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex()
1155 {
1156     unsigned int u0;
1157     asm("call (%0), _optix_read_sbt_gas_idx, ();" : "=r"(u0) :);
1158     return u0;
1159 }
1160
1161 static __forceinline__ __device__ unsigned int optixGetInstanceId()
1162 {
1163     unsigned int u0;
1164     asm("call (%0), _optix_read_instance_id, ();" : "=r"(u0) :);
1165     return u0;
1166 }
1167
1168 static __forceinline__ __device__ unsigned int optixGetInstanceIndex()
1169 {
1170     unsigned int u0;
1171     asm("call (%0), _optix_read_instance_idx, ();" : "=r"(u0) :);
1172     return u0;
1173 }
1174
1175 static __forceinline__ __device__ unsigned int optixGetHitKind()
1176 {
1177     unsigned int u0;
1178     asm("call (%0), _optix_get_hit_kind, ();" : "=r"(u0) :);
1179     return u0;
1180 }
1181
1182 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind)
1183 {
1184     unsigned int u0;
1185     asm("call (%0), _optix_get_primitive_type_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));
1186     return (OptixPrimitiveType)u0;
1187 }
1188
1189 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind)
```

```
1190 {
1191     unsigned int u0;
1192     asm("call (%0), _optix_get_backface_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));
1193     return (u0 == 0x1);
1194 }
1195
1196 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind)
1197 {
1198     return !optixIsBackFaceHit(hitKind);
1199 }
1200
1201
1202 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType()
1203 {
1204     return optixGetPrimitiveType(optixGetHitKind());
1205 }
1206
1207 static __forceinline__ __device__ bool optixIsBackFaceHit()
1208 {
1209     return optixIsBackFaceHit(optixGetHitKind());
1210 }
1211
1212 static __forceinline__ __device__ bool optixIsFrontFaceHit()
1213 {
1214     return optixIsFrontFaceHit(optixGetHitKind());
1215 }
1216
1217 static __forceinline__ __device__ bool optixIsTriangleHit()
1218 {
1219     return optixIsTriangleFrontFaceHit() || optixIsTriangleBackFaceHit();
1220 }
1221
1222 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit()
1223 {
1224     return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE;
1225 }
1226
1227 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit()
1228 {
1229     return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_BACK_FACE;
1230 }
1231
1232 static __forceinline__ __device__ float optixGetCurveParameter()
1233 {
1234     return __int_as_float(optixGetAttribute_0());
1235 }
1236
1237 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics()
1238 {
1239     float f0, f1;
1240     asm("call (%0, %1), _optix_get_triangle_barycentrics, ();" : "=f"(f0), "=f"(f1) :);
1241     return make_float2(f0, f1);
1242 }
1243
1244 static __forceinline__ __device__ uint3 optixGetLaunchIndex()
1245 {
1246     unsigned int u0, u1, u2;
1247     asm("call (%0), _optix_get_launch_index_x, ();" : "=r"(u0) :);
1248     asm("call (%0), _optix_get_launch_index_y, ();" : "=r"(u1) :);
1249     asm("call (%0), _optix_get_launch_index_z, ();" : "=r"(u2) :);
1250     return make_uint3(u0, u1, u2);
1251 }
1252
1253 static __forceinline__ __device__ uint3 optixGetLaunchDimensions()
1254 {
1255     unsigned int u0, u1, u2;
1256     asm("call (%0), _optix_get_launch_dimension_x, ();" : "=r"(u0) :);
```

```
1257     asm("call (%0), _optix_get_launch_dimension_y, ();" : "=r"(u1) :);
1258     asm("call (%0), _optix_get_launch_dimension_z, ();" : "=r"(u2) :);
1259     return make_uint3(u0, u1, u2);
1260 }
1261
1262 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer()
1263 {
1264     unsigned long long ptr;
1265     asm("call (%0), _optix_get_sbt_data_ptr_64, ();" : "=l"(ptr) :);
1266     return (CUdeviceptr)ptr;
1267 }
1268
1269 static __forceinline__ __device__ void optixThrowException(int exceptionCode)
1270 {
1271     asm volatile(
1272         "call _optix_throw_exception_0, (%0);"
1273         : /* no return value */
1274         : "r"(exceptionCode)
1275         :);
1276 }
1277
1278 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0)
1279 {
1280     asm volatile(
1281         "call _optix_throw_exception_1, (%0, %1);"
1282         : /* no return value */
1283         : "r"(exceptionCode), "r"(exceptionDetail0)
1284         :);
1285 }
1286
1287 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1)
1288 {
1289     asm volatile(
1290         "call _optix_throw_exception_2, (%0, %1, %2);"
1291         : /* no return value */
1292         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1)
1293         :);
1294 }
1295
1296 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
1297 {
1298     asm volatile(
1299         "call _optix_throw_exception_3, (%0, %1, %2, %3);"
1300         : /* no return value */
1301         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2)
1302         :);
1303 }
1304
1305 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3)
1306 {
1307     asm volatile(
1308         "call _optix_throw_exception_4, (%0, %1, %2, %3, %4);"
1309         : /* no return value */
1310         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3)
1311         :);
1312 }
1313
1314 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3, unsigned int exceptionDetail4)
1315 {
```

```
1316     asm volatile(
1317         "call _optix_throw_exception_5, (%0, %1, %2, %3, %4, %5);"
1318         : /* no return value */
1319         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4)
1320         :);
1321 }
1322
1323 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)
1324 {
1325     asm volatile(
1326         "call _optix_throw_exception_6, (%0, %1, %2, %3, %4, %5, %6);"
1327         : /* no return value */
1328         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5)
1329         :);
1330 }
1331
1332 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)
1333 {
1334     asm volatile(
1335         "call _optix_throw_exception_7, (%0, %1, %2, %3, %4, %5, %6, %7);"
1336         : /* no return value */
1337         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6)
1338         :);
1339 }
1340
1341 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)
1342 {
1343     asm volatile(
1344         "call _optix_throw_exception_8, (%0, %1, %2, %3, %4, %5, %6, %7, %8);"
1345         : /* no return value */
1346         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6),
"r"(exceptionDetail7)
1347         :);
1348 }
1349
1350 static __forceinline__ __device__ int optixGetExceptionCode( )
1351 {
1352     int s0;
1353     asm("call (%0), _optix_get_exception_code, ();" : "=r"(s0) :);
1354     return s0;
1355 }
1356
1357 #define OPTIX_DEFINE_optixGetExceptionDetail_BODY(which)
\
1358 unsigned int ret;
\
1359 asm("call (%0), _optix_get_exception_detail_" #which ", ();" : "=r"(ret) :);
\
1360     return ret;
1361
1362 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0( )
1363 {
1364     OPTIX_DEFINE_optixGetExceptionDetail_BODY(0);
1365 }
1366
```

```
1367 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1()
1368 {
1369     OPTIX_DEFINE_optixGetExceptionDetail_BODY(1);
1370 }
1371
1372 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2()
1373 {
1374     OPTIX_DEFINE_optixGetExceptionDetail_BODY(2);
1375 }
1376
1377 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3()
1378 {
1379     OPTIX_DEFINE_optixGetExceptionDetail_BODY(3);
1380 }
1381
1382 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4()
1383 {
1384     OPTIX_DEFINE_optixGetExceptionDetail_BODY(4);
1385 }
1386
1387 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5()
1388 {
1389     OPTIX_DEFINE_optixGetExceptionDetail_BODY(5);
1390 }
1391
1392 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6()
1393 {
1394     OPTIX_DEFINE_optixGetExceptionDetail_BODY(6);
1395 }
1396
1397 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7()
1398 {
1399     OPTIX_DEFINE_optixGetExceptionDetail_BODY(7);
1400 }
1401
1402 #undef OPTIX_DEFINE_optixGetExceptionDetail_BODY
1403
1404 static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable()
1405 {
1406     unsigned long long handle;
1407     asm("call (%0), _optix_get_exception_invalid_traversable, ();" : "=l"(handle) :);
1408     return (OptixTraversableHandle)handle;
1409 }
1410
1411 static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset()
1412 {
1413     int s0;
1414     asm("call (%0), _optix_get_exception_invalid_sbt_offset, ();" : "=r"(s0) :);
1415     return s0;
1416 }
1417
1418 static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay()
1419 {
1420     float rayOriginX, rayOriginY, rayOriginZ, rayDirectionX, rayDirectionY, rayDirectionZ, tmin, tmax,
rayTime;
1421     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_exception_invalid_ray, ();"
1422         : "=f"(rayOriginX), "=f"(rayOriginY), "=f"(rayOriginZ), "=f"(rayDirectionX),
"=f"(rayDirectionY),
1423           "=f"(rayDirectionZ), "=f"(tmin), "=f"(tmax), "=f"(rayTime)
1424         :);
1425     OptixInvalidRayExceptionDetails ray;
1426     ray.origin    = make_float3(rayOriginX, rayOriginY, rayOriginZ);
1427     ray.direction = make_float3(rayDirectionX, rayDirectionY, rayDirectionZ);
1428     ray.tmin      = tmin;
1429     ray.tmax      = tmax;
1430     ray.time      = rayTime;
1431     return ray;
```

```
1432 }
1433
1434 static __forceinline__ __device__ OptixParameterMismatchExceptionDetails
optixGetExceptionParameterMismatch()
1435 {
1436     unsigned int expected, actual, sbtIdx;
1437     unsigned long long calleeName;
1438     asm(
1439         "call (%0, %1, %2, %3), _optix_get_exception_parameter_mismatch, ();"
1440         : "=r"(expected), "=r"(actual), "=r"(sbtIdx), "=l"(calleeName) :);
1441     OptixParameterMismatchExceptionDetails details;
1442     details.expectedParameterCount = expected;
1443     details.passedArgumentCount = actual;
1444     details.sbtIndex = sbtIdx;
1445     details.callableName = (char*)calleeName;
1446     return details;
1447 }
1448
1449 static __forceinline__ __device__ char* optixGetExceptionLineInfo()
1450 {
1451     unsigned long long ptr;
1452     asm("call (%0), _optix_get_exception_line_info, ();" : "=l"(ptr) :);
1453     return (char*)ptr;
1454 }
1455
1456 template <typename ReturnT, typename... ArgTypes>
1457 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args)
1458 {
1459     unsigned long long func;
1460     asm("call (%0), _optix_call_direct_callable,(%1);" : "=l"(func) : "r"(sbtIndex) :);
1461     using funcT = ReturnT (*)(ArgTypes...);
1462     funcT call  = (funcT)(func);
1463     return call(args...);
1464 }
1465
1466 template <typename ReturnT, typename... ArgTypes>
1467 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes... args)
1468 {
1469     unsigned long long func;
1470     asm("call (%0), _optix_call_continuation_callable,(%1);" : "=l"(func) : "r"(sbtIndex) :);
1471     using funcT = ReturnT (*)(ArgTypes...);
1472     funcT call  = (funcT)(func);
1473     return call(args...);
1474 }
1475 #endif
1476
1477 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int
texInfo, float x, float y, unsigned int* singleMipLevel)
1478 {
1479     uint4              result;
1480     unsigned long long resultPtr         = reinterpret_cast<unsigned long long>(&result);
1481     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
1482     // Cast float args to integers, because the intrinics take .b32 arguments when compiled to PTX.
1483     asm volatile(
1484         "call _optix_tex_footprint_2d_v2"
1485         ", (%0, %1, %2, %3, %4, %5);"
1486         :
1487         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
1488           "l"(singleMipLevelPtr), "l"(resultPtr)
1489         :);
1490     return result;
1491 }
1492
1493 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
1494                                                                 unsigned int       texInfo,
1495                                                                 float              x,
1496                                                                 float              y,
```

```
1497                                                                    float              dPdx_x,
1498                                                                    float              dPdx_y,
1499                                                                    float              dPdy_x,
1500                                                                    float              dPdy_y,
1501                                                                    bool               coarse,
1502                                                                    unsigned int*      singleMipLevel)
1503 {
1504     uint4              result;
1505     unsigned long long resultPtr         = reinterpret_cast<unsigned long long>(&result);
1506     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
1507     // Cast float args to integers, because the intrinics take .b32 arguments when compiled to PTX.
1508     asm volatile(
1509         "call _optix_tex_footprint_2d_grad_v2"
1510         ", (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
1511         :
1512         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
1513           "r"(__float_as_uint(dPdx_x)), "r"(__float_as_uint(dPdx_y)), "r"(__float_as_uint(dPdy_x)),
1514           "r"(__float_as_uint(dPdy_y)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
"l"(resultPtr)
1515         :);
1516
1517     return result;
1518 }
1519
1520 static __forceinline__ __device__ uint4
1521 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool
coarse, unsigned int* singleMipLevel)
1522 {
1523     uint4              result;
1524     unsigned long long resultPtr         = reinterpret_cast<unsigned long long>(&result);
1525     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
1526     // Cast float args to integers, because the intrinics take .b32 arguments when compiled to PTX.
1527     asm volatile(
1528         "call _optix_tex_footprint_2d_lod_v2"
1529         ", (%0, %1, %2, %3, %4, %5, %6, %7);"
1530         :
1531         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
1532           "r"(__float_as_uint(level)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
"l"(resultPtr)
1533         :);
1534     return result;
1535 }
```

8.3 optix_7_device_impl_exception.h File Reference

Namespaces

- namespace optix_impl

Functions

- static __forceinline__ __device__ void optix_impl::optixDumpStaticTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optix_impl ::optixDumpMotionMatrixTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optix_impl::optixDumpSrtMatrixTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optix_impl::optixDumpInstanceFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optix_impl::optixDumpTransform (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optix_impl::optixDumpTransformList ()
- static __forceinline__ __device__ void optix_impl::optixDumpExceptionDetails ()

8.3.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Device side implementation for exception helper function.

8.4 optix_7_device_impl_exception.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
3 *
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly
8 * prohibited.
9 *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
29 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
30 #error("optix_7_device_impl_exception.h is an internal header file and must not be used directly.  Please
use optix_device.h or optix.h instead.")
31 #endif
32
33 #ifndef __optix_optix_7_device_impl_exception_h__
34 #define __optix_optix_7_device_impl_exception_h__
35
36 #if !defined(__CUDACC_RTC__)
37 #include <cstdio> /* for printf */
38 #endif
39
40 namespace optix_impl {
41
42     static __forceinline__ __device__ void optixDumpStaticTransformFromHandle(OptixTraversableHandle
handle)
43     {
44         const OptixStaticTransform* traversable = optixGetStaticTransformFromHandle(handle);
45         if(traversable)
46         {
47             const uint3 index = optixGetLaunchIndex();
48             printf("(%4i,%4i,%4i)     OptixStaticTransform@%p = {\n"
49                    "                          child        = %p,\n"
50                    "                          transform    = { %f,%f,%f,%f,\n"
51                    "                                           %f,%f,%f,%f,\n"
52                    "                                           %f,%f,%f,%f } }\n",
53                 index.x,index.y,index.z,
54                 traversable,
55                 (void*)traversable->child,
56                 traversable->transform[0], traversable->transform[1], traversable->transform[2],
traversable->transform[3],
57                 traversable->transform[4], traversable->transform[5], traversable->transform[6],
traversable->transform[7],
```

```
58                 traversable->transform[8], traversable->transform[9], traversable->transform[10],
traversable->transform[11]);
59         }
60     }
61
62     static __forceinline__ __device__ void
optixDumpMotionMatrixTransformFromHandle(OptixTraversableHandle handle)
63     {
64         const OptixMatrixMotionTransform* traversable =  optixGetMatrixMotionTransformFromHandle(handle);
65         if(traversable)
66         {
67             const uint3 index = optixGetLaunchIndex();
68             printf("(%4i,%4i,%4i)     OptixMatrixMotionTransform@%p = {\n"
69                    "                          child         = %p,\n"
70                    "                          motionOptions = { numKeys = %i, flags = %i, timeBegin = %f,
timeEnd = %f },\n"
71                    "                          transform     = { { %f,%f,%f,%f,\n"
72                    "                                            %f,%f,%f,%f,\n"
73                    "                                            %f,%f,%f,%f }, ... }\n",
74                 index.x,index.y,index.z,
75                 traversable,
76                 (void*)traversable->child,
77                 (int)traversable->motionOptions.numKeys, (int)traversable->motionOptions.flags,
traversable->motionOptions.timeBegin, traversable->motionOptions.timeEnd,
78                 traversable->transform[0][0], traversable->transform[0][1], traversable->transform[0][2],
traversable->transform[0][3],
79                 traversable->transform[0][4], traversable->transform[0][5], traversable->transform[0][6],
traversable->transform[0][7],
80                traversable->transform[0][8], traversable->transform[0][9], traversable->transform[0][10],
traversable->transform[0][11]);
81         }
82     }
83
84     static __forceinline__ __device__ void optixDumpSrtMatrixTransformFromHandle(OptixTraversableHandle
handle)
85     {
86         const OptixSRTMotionTransform* traversable =  optixGetSRTMotionTransformFromHandle(handle);
87         if(traversable)
88         {
89             const uint3 index = optixGetLaunchIndex();
90             printf("(%4i,%4i,%4i)     OptixSRTMotionTransform@%p = {\n"
91                    "                          child         = %p,\n"
92                    "                          motionOptions = { numKeys = %i, flags = %i, timeBegin = %f,
timeEnd = %f },\n"
93                    "                          srtData       = { { sx  = %f,  a = %f,   b = %f, pvx = %f,\n"
94                    "                                            sy  = %f,  c = %f, pvy = %f,  sz = %f,\n"
95                    "                                            pvz = %f, qx = %f,  qy = %f,  qz = %f,\n"
96                    "                                          qw  = %f, tx = %f,  ty = %f,  tz = %f }, ... }\n",
97                 index.x,index.y,index.z,
98                 traversable,
99                 (void*)traversable->child,
100                 (int)traversable->motionOptions.numKeys, (int)traversable->motionOptions.flags,
traversable->motionOptions.timeBegin, traversable->motionOptions.timeEnd,
101                 traversable->srtData[0].sx, traversable->srtData[0].a, traversable->srtData[0].b,
traversable->srtData[0].pvx,
102                 traversable->srtData[0].sy, traversable->srtData[0].c,
traversable->srtData[0].pvy,traversable->srtData[0].sz,
103                 traversable->srtData[0].pvz,traversable->srtData[0].qx,traversable->srtData[0].qy,
traversable->srtData[0].qz,
104                 traversable->srtData[0].qw, traversable->srtData[0].tx,traversable->srtData[0].ty,
traversable->srtData[0].tz);
105         }
106     }
107
108     static __forceinline__ __device__ void optixDumpInstanceFromHandle(OptixTraversableHandle handle)
109     {
110         if(optixGetTransformTypeFromHandle(handle) == OPTIX_TRANSFORM_TYPE_INSTANCE)
```

```
111         {
112             unsigned int instanceId = optixGetInstanceIdFromHandle(handle);
113             const float4* transform = optixGetInstanceTransformFromHandle(handle);
114 
115             const uint3 index = optixGetLaunchIndex();
116             printf("(%4i,%4i,%4i)     OptixInstance = {\n"
117                    "                      instanceId = %i,\n"
118                    "                      transform  = { %f,%f,%f,%f,\n"
119                    "                                     %f,%f,%f,%f,\n"
120                    "                                     %f,%f,%f,%f } }\n",
121                 index.x,index.y,index.z,
122                 instanceId,
123                 transform[0].x, transform[0].y, transform[0].z,  transform[0].w,
124                 transform[1].x, transform[1].y, transform[1].z,  transform[1].w,
125                 transform[2].x, transform[2].y, transform[2].z,  transform[2].w);
126         }
127     }
128 
129     static __forceinline__ __device__ void optixDumpTransform(OptixTraversableHandle handle)
130     {
131         const OptixTransformType type = optixGetTransformTypeFromHandle(handle);
132         const uint3 index = optixGetLaunchIndex();
133 
134         switch(type)
135         {
136             case OPTIX_TRANSFORM_TYPE_NONE:
137                 break;
138             case OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM:
139                 optixDumpStaticTransformFromHandle(handle);
140                 break;
141             case OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM:
142                 optixDumpMotionMatrixTransformFromHandle(handle);
143                 break;
144             case OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM:
145                 optixDumpSrtMatrixTransformFromHandle(handle);
146                 break;
147             case OPTIX_TRANSFORM_TYPE_INSTANCE:
148                 optixDumpInstanceFromHandle(handle);
149                 break;
150             default:
151                 break;
152         }
153     }
154 
155     static __forceinline__ __device__ void optixDumpTransformList()
156     {
157         const int tlistSize = optixGetTransformListSize();
158         const uint3 index = optixGetLaunchIndex();
159 
160         printf("(%4i,%4i,%4i) transform list of size %i:\n", index.x,index.y,index.z, tlistSize);
161 
162         for(unsigned int i = 0 ; i < tlistSize ; ++i)
163         {
164             OptixTraversableHandle handle = optixGetTransformListHandle(i);
165             printf("(%4i,%4i,%4i)   transform[%i] = %p\n", index.x, index.y, index.z, i, (void*)handle);
166             optixDumpTransform(handle);
167         }
168     }
169 
170     static __forceinline__ __device__ void optixDumpExceptionDetails()
171     {
172         bool dumpTlist = false;
173         const int exceptionCode = optixGetExceptionCode();
174         const uint3 index = optixGetLaunchIndex();
175 
176         if(exceptionCode == OPTIX_EXCEPTION_CODE_STACK_OVERFLOW)
177         {
```

```
178             printf("(%4i,%4i,%4i) error: stack overflow\n", index.x,index.y,index.z);
179         }
180         else if(exceptionCode == OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED)
181         {
182             printf("(%4i,%4i,%4i) error: trace depth exceeded\n", index.x,index.y,index.z);
183         }
184         else if(exceptionCode == OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED)
185         {
186             printf("(%4i,%4i,%4i) error: traversal depth exceeded\n", index.x,index.y,index.z);
187             dumpTlist = true;
188         }
189         else if(exceptionCode == OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE)
190         {
191             OptixTraversableHandle handle = optixGetExceptionInvalidTraversable();
192             printf("(%4i,%4i,%4i) error: invalid traversable %p\n", index.x,index.y,index.z,
(void*)handle);
193             dumpTlist = true;
194         }
195         else if(exceptionCode == OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT)
196         {
197             int sbtOffset = optixGetExceptionInvalidSbtOffset();
198             printf("(%4i,%4i,%4i) error: invalid miss sbt of %i\n", index.x,index.y,index.z, sbtOffset);
199         }
200         else if(exceptionCode == OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT)
201         {
202             int sbtOffset = optixGetExceptionInvalidSbtOffset();
203             printf("(%4i,%4i,%4i) error: invalid hit sbt of %i at primitive with gas sbt index %i\n",
index.x,index.y,index.z, sbtOffset, optixGetSbtGASIndex());
204             dumpTlist = true;
205         }
206         else if(exceptionCode == OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE)
207         {
208             dumpTlist = true;
209             printf("(%4i,%4i,%4i) error: shader encountered unsupported builtin type\n"
210                    "       call location:   %s\n", index.x, index.y, index.z,
optixGetExceptionLineInfo());
211         }
212         else if(exceptionCode == OPTIX_EXCEPTION_CODE_INVALID_RAY)
213         {
214             OptixInvalidRayExceptionDetails ray = optixGetExceptionInvalidRay();
215             printf("(%4i,%4i,%4i) error: encountered an invalid ray:\n", index.x, index.y, index.z);
216             printf(
217                 "       origin:           [%f, %f, %f]\n"
218                 "       direction:        [%f, %f, %f]\n"
219                 "       tmin:              %f\n"
220                 "       tmax:              %f\n"
221                 "       rayTime:           %f\n"
222                 "       call location:   %s\n",
223                ray.origin.x, ray.origin.y, ray.origin.z, ray.direction.x, ray.direction.y,
224                ray.direction.z, ray.tmin, ray.tmax, ray.time, optixGetExceptionLineInfo());
225         }
226         else if(exceptionCode == OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH)
227         {
228              OptixParameterMismatchExceptionDetails details = optixGetExceptionParameterMismatch();
229              printf("(%4i,%4i,%4i) error: parameter mismatch in callable call.\n", index.x, index.y,
index.z);
230              printf(
231                 "       passed packed arguments:        %u 32 Bit values\n"
232                 "       expected packed parameters:     %u 32 Bit values\n"
233                 "       SBT index:                        %u\n"
234                 "       called function:                 %s\n"
235                 "       call location:                    %s\n",
236                details.passedArgumentCount, details.expectedParameterCount, details.sbtIndex,
237                details.callableName, optixGetExceptionLineInfo());
238         }
239         else if(exceptionCode == OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH)
240         {
```

```
241             dumpTlist = true;
242             printf("(%4i,%4i,%4i) error: mismatch between builtin IS shader and build input\n"
243                    "       call location:   %s\n", index.x,index.y,index.z, optixGetExceptionLineInfo());
244         }
245         else if(exceptionCode == OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT)
246         {
247             int sbtOffset = optixGetExceptionInvalidSbtOffset();
248             printf("(%4i,%4i,%4i) error: invalid sbt offset of %i for callable program\n", index.x,
index.y, index.z, sbtOffset);
249         }
250         else if(exceptionCode == OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD)
251         {
252             int sbtOffset = optixGetExceptionInvalidSbtOffset();
253             printf("(%4i,%4i,%4i) error: invalid sbt offset of %i for direct callable program\n",
index.x, index.y, index.z, sbtOffset);
254         }
255         else if(exceptionCode == OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD)
256         {
257             int sbtOffset = optixGetExceptionInvalidSbtOffset();
258             printf("(%4i,%4i,%4i) error: invalid sbt offset of %i for continuation callable program\n",
index.x, index.y, index.z, sbtOffset);
259         }
260         else if(exceptionCode == OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS)
261         {
262             OptixTraversableHandle handle = optixGetExceptionInvalidTraversable();
263             printf("(%4i,%4i,%4i) error: unsupported single GAS traversable graph %p\n",
index.x,index.y,index.z, (void*)handle);
264             dumpTlist = true;
265         }
266         else if((exceptionCode <= OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0) && (exceptionCode >=
OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2))
267         {
268             printf("(%4i,%4i,%4i) error: invalid value for argument %i\n", index.x,index.y,index.z,
-(exceptionCode - OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0));
269         }
270         else if(exceptionCode == OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS)
271         {
272             printf("(%4i,%4i,%4i) error: unsupported random data access\n", index.x,index.y,index.z);
273         }
274         else if(exceptionCode == OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH)
275         {
276             printf("(%4i,%4i,%4i) error: payload type mismatch between program and optixTrace call\n",
index.x,index.y,index.z);
277         }
278         else if(exceptionCode >= 0)
279         {
280             dumpTlist = true;
281             printf("(%4i,%4i,%4i) error: user exception with error code %i\n"
282                    "       call location:   %s\n", index.x, index.y, index.z, exceptionCode,
optixGetExceptionLineInfo());
283         }
284         else
285         {
286             printf("(%4i,%4i,%4i) error: unknown exception with error code %i\n",
index.x,index.y,index.z, exceptionCode);
287         }
288
289         if(dumpTlist)
290             optixDumpTransformList();
291     }
292
293 }  // namespace optix_impl
294
295 #endif
```

8.5 optix_7_device_impl_transformations.h File Reference

Namespaces

- namespace optix_impl

Functions

- static __forceinline__ __device__ float4 optix_impl::optixAddFloat4 (const float4 &a, const float4 &b)
- static __forceinline__ __device__ float4 optix_impl::optixMulFloat4 (const float4 &a, float b)
- static __forceinline__ __device__ uint4 optix_impl::optixLdg (unsigned long long addr)
- template<class T >
  static __forceinline__ __device__ T optix_impl::optixLoadReadOnlyAlign16 (const T *ptr)
- static __forceinline__ __device__ float4 optix_impl::optixMultiplyRowMatrix (const float4 vec, const float4 m0, const float4 m1, const float4 m2)
- static __forceinline__ __device__ void optix_impl::optixGetMatrixFromSrt (float4 &m0, float4 &m1, float4 &m2, const OptixSRTData &srt)
- static __forceinline__ __device__ void optix_impl::optixInvertMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ void optix_impl::optixLoadInterpolatedMatrixKey (float4 &m0, float4 &m1, float4 &m2, const float4 *matrix, const float t1)
- static __forceinline__ __device__ void optix_impl::optixLoadInterpolatedSrtKey (float4 &srt0, float4 &srt1, float4 &srt2, float4 &srt3, const float4 *srt, const float t1)
- static __forceinline__ __device__ void optix_impl::optixResolveMotionKey (float &localt, int &key, const OptixMotionOptions &options, const float globalt)
- static __forceinline__ __device__ void optix_impl::optixGetInterpolatedTransformation (float4 &trf0, float4 &trf1, float4 &trf2, const OptixMatrixMotionTransform *transformData, const float time)
- static __forceinline__ __device__ void optix_impl::optixGetInterpolatedTransformation (float4 &trf0, float4 &trf1, float4 &trf2, const OptixSRTMotionTransform *transformData, const float time)
- static __forceinline__ __device__ void optix_impl ::optixGetInterpolatedTransformationFromHandle (float4 &trf0, float4 &trf1, float4 &trf2, const OptixTraversableHandle handle, const float time, const bool objectToWorld)
- static __forceinline__ __device__ void optix_impl::optixGetWorldToObjectTransformMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ void optix_impl::optixGetObjectToWorldTransformMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ float3 optix_impl::optixTransformPoint (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &p)
- static __forceinline__ __device__ float3 optix_impl::optixTransformVector (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &v)
- static __forceinline__ __device__ float3 optix_impl::optixTransformNormal (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &n)

8.5.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

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

8.6 optix_7_device_impl_transformations.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
3 *
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly
8 * prohibited.
9 *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
29 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
30 #error("optix_7_device_impl_transformations.h is an internal header file and must not be used directly.
Please use optix_device.h or optix.h instead.")
31 #endif
32
33 #ifndef __optix_optix_7_device_impl_transformations_h__
34 #define __optix_optix_7_device_impl_transformations_h__
35
36 namespace optix_impl {
37
38 static __forceinline__ __device__ float4 optixAddFloat4(const float4& a, const float4& b)
39 {
40     return make_float4(a.x + b.x, a.y + b.y, a.z + b.z, a.w + b.w);
41 }
42
43 static __forceinline__ __device__ float4 optixMulFloat4(const float4& a, float b)
44 {
45     return make_float4(a.x * b, a.y * b, a.z * b, a.w * b);
46 }
47
48 static __forceinline__ __device__ uint4 optixLdg(unsigned long long addr)
49 {
50     const uint4* ptr;
51     asm volatile("cvta.to.global.u64 %0, %1;" : "=l"(ptr) : "l"(addr));
52     uint4 ret;
53     asm volatile("ld.global.v4.u32 {%0,%1,%2,%3}, [%4];"
54                  : "=r"(ret.x), "=r"(ret.y), "=r"(ret.z), "=r"(ret.w)
55                  : "l"(ptr));
56     return ret;
57 }
58
59 template <class T>
60 static __forceinline__ __device__ T optixLoadReadOnlyAlign16(const T* ptr)
61 {
62     T v;
63     for(int ofs                     = 0; ofs < sizeof(T); ofs += 16)
64         *(uint4*)((char*)&v + ofs) = optixLdg((unsigned long long)((char*)ptr + ofs));
65     return v;
66 }
67
68 // Multiplies the row vector vec with the 3x4 matrix with rows m0, m1, and m2
69 static __forceinline__ __device__ float4 optixMultiplyRowMatrix(const float4 vec, const float4 m0, const
float4 m1, const float4 m2)
70 {
```

```
71     float4 result;
72
73     result.x = vec.x * m0.x + vec.y * m1.x + vec.z * m2.x;
74     result.y = vec.x * m0.y + vec.y * m1.y + vec.z * m2.y;
75     result.z = vec.x * m0.z + vec.y * m1.z + vec.z * m2.z;
76     result.w = vec.x * m0.w + vec.y * m1.w + vec.z * m2.w + vec.w;
77
78     return result;
79 }
80
81 // Converts the SRT transformation srt into a 3x4 matrix with rows m0, m1, and m2
82 static __forceinline__ __device__ void optixGetMatrixFromSrt(float4& m0, float4& m1, float4& m2, const
OptixSRTData& srt)
83 {
84     const float4 q = {srt.qx, srt.qy, srt.qz, srt.qw};
85
86     // normalize
87     const float  inv_sql = 1.f / (srt.qx * srt.qx + srt.qy * srt.qy + srt.qz * srt.qz + srt.qw * srt.qw);
88     const float4 nq      = optixMulFloat4(q, inv_sql);
89
90     const float sqw = q.w * nq.w;
91     const float sqx = q.x * nq.x;
92     const float sqy = q.y * nq.y;
93     const float sqz = q.z * nq.z;
94
95     const float xy = q.x * nq.y;
96     const float zw = q.z * nq.w;
97     const float xz = q.x * nq.z;
98     const float yw = q.y * nq.w;
99     const float yz = q.y * nq.z;
100     const float xw = q.x * nq.w;
101
102     m0.x = (sqx - sqy - sqz + sqw);
103     m0.y = 2.0f * (xy - zw);
104     m0.z = 2.0f * (xz + yw);
105
106     m1.x = 2.0f * (xy + zw);
107     m1.y = (-sqx + sqy - sqz + sqw);
108     m1.z = 2.0f * (yz - xw);
109
110     m2.x = 2.0f * (xz - yw);
111     m2.y = 2.0f * (yz + xw);
112     m2.z = (-sqx - sqy + sqz + sqw);
113
114     m0.w = m0.x * srt.pvx + m0.y * srt.pvy + m0.z * srt.pvz + srt.tx;
115     m1.w = m1.x * srt.pvx + m1.y * srt.pvy + m1.z * srt.pvz + srt.ty;
116     m2.w = m2.x * srt.pvx + m2.y * srt.pvy + m2.z * srt.pvz + srt.tz;
117
118     m0.z = m0.x * srt.b + m0.y * srt.c + m0.z * srt.sz;
119     m1.z = m1.x * srt.b + m1.y * srt.c + m1.z * srt.sz;
120     m2.z = m2.x * srt.b + m2.y * srt.c + m2.z * srt.sz;
121
122     m0.y = m0.x * srt.a + m0.y * srt.sy;
123     m1.y = m1.x * srt.a + m1.y * srt.sy;
124     m2.y = m2.x * srt.a + m2.y * srt.sy;
125
126     m0.x = m0.x * srt.sx;
127     m1.x = m1.x * srt.sx;
128     m2.x = m2.x * srt.sx;
129 }
130
131 // Inverts a 3x4 matrix in place
132 static __forceinline__ __device__ void optixInvertMatrix(float4& m0, float4& m1, float4& m2)
133 {
134     const float det3 =
135         m0.x * (m1.y * m2.z - m1.z * m2.y) - m0.y * (m1.x * m2.z - m1.z * m2.x) + m0.z * (m1.x * m2.y -
m1.y * m2.x);
```

```
136
137     const float inv_det3 = 1.0f / det3;
138
139     float inv3[3][3];
140     inv3[0][0] = inv_det3 * (m1.y * m2.z - m2.y * m1.z);
141     inv3[0][1] = inv_det3 * (m0.z * m2.y - m2.z * m0.y);
142     inv3[0][2] = inv_det3 * (m0.y * m1.z - m1.y * m0.z);
143
144     inv3[1][0] = inv_det3 * (m1.z * m2.x - m2.z * m1.x);
145     inv3[1][1] = inv_det3 * (m0.x * m2.z - m2.x * m0.z);
146     inv3[1][2] = inv_det3 * (m0.z * m1.x - m1.z * m0.x);
147
148     inv3[2][0] = inv_det3 * (m1.x * m2.y - m2.x * m1.y);
149     inv3[2][1] = inv_det3 * (m0.y * m2.x - m2.y * m0.x);
150     inv3[2][2] = inv_det3 * (m0.x * m1.y - m1.x * m0.y);
151
152     const float b[3] = {m0.w, m1.w, m2.w};
153
154     m0.x = inv3[0][0];
155     m0.y = inv3[0][1];
156     m0.z = inv3[0][2];
157     m0.w = -inv3[0][0] * b[0] - inv3[0][1] * b[1] - inv3[0][2] * b[2];
158
159     m1.x = inv3[1][0];
160     m1.y = inv3[1][1];
161     m1.z = inv3[1][2];
162     m1.w = -inv3[1][0] * b[0] - inv3[1][1] * b[1] - inv3[1][2] * b[2];
163
164     m2.x = inv3[2][0];
165     m2.y = inv3[2][1];
166     m2.z = inv3[2][2];
167     m2.w = -inv3[2][0] * b[0] - inv3[2][1] * b[1] - inv3[2][2] * b[2];
168 }
169
170 static __forceinline__ __device__ void optixLoadInterpolatedMatrixKey(float4& m0, float4& m1, float4&
m2, const float4* matrix, const float t1)
171 {
172     m0 = optixLoadReadOnlyAlign16(&matrix[0]);
173     m1 = optixLoadReadOnlyAlign16(&matrix[1]);
174     m2 = optixLoadReadOnlyAlign16(&matrix[2]);
175
176     // The conditional prevents concurrent loads leading to spills
177     if(t1 > 0.0f)
178     {
179         const float t0 = 1.0f - t1;
180         m0 = optixAddFloat4(optixMulFloat4(m0, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[3]),
t1));
181         m1 = optixAddFloat4(optixMulFloat4(m1, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[4]),
t1));
182         m2 = optixAddFloat4(optixMulFloat4(m2, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[5]),
t1));
183     }
184 }
185
186 static __forceinline__ __device__ void optixLoadInterpolatedSrtKey(float4&       srt0,
187                                                                     float4&       srt1,
188                                                                     float4&       srt2,
189                                                                     float4&       srt3,
190                                                                     const float4* srt,
191                                                                     const float   t1)
192 {
193     srt0 = optixLoadReadOnlyAlign16(&srt[0]);
194     srt1 = optixLoadReadOnlyAlign16(&srt[1]);
195     srt2 = optixLoadReadOnlyAlign16(&srt[2]);
196     srt3 = optixLoadReadOnlyAlign16(&srt[3]);
197
198     // The conditional prevents concurrent loads leading to spills
```

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

```
253                                                                 float4&                     trf2,
254                                                                     const OptixSRTMotionTransform*
transformData,
255                                                                 const float                 time)
256 {
257     // Compute key and intra key time
258     float keyTime;
259     int   key;
260     optixResolveMotionKey(keyTime, key, optixLoadReadOnlyAlign16(transformData).motionOptions, time);
261
262     // Get pointer to left key
263     const float4* dataPtr = reinterpret_cast<const float4*>(&transformData->srtData[key]);
264
265     // Load and interpolated SRT keys
266     float4 data[4];
267     optixLoadInterpolatedSrtKey(data[0], data[1], data[2], data[3], dataPtr, keyTime);
268
269     OptixSRTData srt = {data[0].x, data[0].y, data[0].z, data[0].w, data[1].x, data[1].y, data[1].z,
data[1].w,
270                         data[2].x, data[2].y, data[2].z, data[2].w, data[3].x, data[3].y, data[3].z,
data[3].w};
271
272     // Convert SRT into a matrix
273     optixGetMatrixFromSrt(trf0, trf1, trf2, srt);
274 }
275
276 // Returns the interpolated transformation matrix for a particular traversable handle and point in time.
277 static __forceinline__ __device__ void optixGetInterpolatedTransformationFromHandle(float4&
trf0,
278                                                                                      float4&
trf1,
279                                                                                      float4&
trf2,
280                                                                                      const
OptixTraversableHandle handle,
281                                                                                      const float
time,
282                                                                                    const bool objectToWorld)
283 {
284     const OptixTransformType type = optixGetTransformTypeFromHandle(handle);
285
286     if(type == OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM || type ==
OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM)
287     {
288         if(type == OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM)
289         {
290             const OptixMatrixMotionTransform* transformData =
optixGetMatrixMotionTransformFromHandle(handle);
291             optixGetInterpolatedTransformation(trf0, trf1, trf2, transformData, time);
292         }
293         else
294         {
295             const OptixSRTMotionTransform* transformData = optixGetSRTMotionTransformFromHandle(handle);
296             optixGetInterpolatedTransformation(trf0, trf1, trf2, transformData, time);
297         }
298
299         if(!objectToWorld)
300             optixInvertMatrix(trf0, trf1, trf2);
301     }
302     else if(type == OPTIX_TRANSFORM_TYPE_INSTANCE || type == OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM)
303     {
304         const float4* transform;
305
306         if(type == OPTIX_TRANSFORM_TYPE_INSTANCE)
307         {
308             transform = (objectToWorld) ? optixGetInstanceTransformFromHandle(handle) :
309                                           optixGetInstanceInverseTransformFromHandle(handle);
```

```
310         }
311         else
312         {
313             const OptixStaticTransform* traversable = optixGetStaticTransformFromHandle(handle);
314             transform = (const float4*)((objectToWorld) ? traversable->transform :
traversable->invTransform);
315         }
316
317         trf0 = optixLoadReadOnlyAlign16(&transform[0]);
318         trf1 = optixLoadReadOnlyAlign16(&transform[1]);
319         trf2 = optixLoadReadOnlyAlign16(&transform[2]);
320     }
321     else
322     {
323         trf0 = {1.0f, 0.0f, 0.0f, 0.0f};
324         trf1 = {0.0f, 1.0f, 0.0f, 0.0f};
325         trf2 = {0.0f, 0.0f, 1.0f, 0.0f};
326     }
327 }
328
329 // Returns the world-to-object transformation matrix resulting from the current transform stack and
current ray time.
330 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float4& m0, float4& m1,
float4& m2)
331 {
332     const unsigned int size = optixGetTransformListSize();
333     const float        time = optixGetRayTime();
334
335 #pragma unroll 1
336     for(unsigned int i = 0; i < size; ++i)
337     {
338         OptixTraversableHandle handle = optixGetTransformListHandle(i);
339
340         float4 trf0, trf1, trf2;
341         optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
false);
342
343         if(i == 0)
344         {
345             m0 = trf0;
346             m1 = trf1;
347             m2 = trf2;
348         }
349         else
350         {
351             // m := trf * m
352             float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
353             m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
354             m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
355             m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
356         }
357     }
358 }
359
360 // Returns the object-to-world transformation matrix resulting from the current transform stack and
current ray time.
361 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float4& m0, float4& m1,
float4& m2)
362 {
363     const int   size = optixGetTransformListSize();
364     const float time = optixGetRayTime();
365
366 #pragma unroll 1
367     for(int i = size - 1; i >= 0; --i)
368     {
369         OptixTraversableHandle handle = optixGetTransformListHandle(i);
370
```

```
371         float4 trf0, trf1, trf2;
372         optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
true);
373 
374         if(i == size - 1)
375         {
376             m0 = trf0;
377             m1 = trf1;
378             m2 = trf2;
379         }
380         else
381         {
382             // m := trf * m
383             float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
384             m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
385             m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
386             m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
387         }
388     }
389 }
390 
391 // Multiplies the 3x4 matrix with rows m0, m1, m2 with the point p.
392 static __forceinline__ __device__ float3 optixTransformPoint(const float4& m0, const float4& m1, const
float4& m2, const float3& p)
393 {
394     float3 result;
395     result.x = m0.x * p.x + m0.y * p.y + m0.z * p.z + m0.w;
396     result.y = m1.x * p.x + m1.y * p.y + m1.z * p.z + m1.w;
397     result.z = m2.x * p.x + m2.y * p.y + m2.z * p.z + m2.w;
398     return result;
399 }
400 
401 // Multiplies the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the vector v.
402 static __forceinline__ __device__ float3 optixTransformVector(const float4& m0, const float4& m1, const
float4& m2, const float3& v)
403 {
404     float3 result;
405     result.x = m0.x * v.x + m0.y * v.y + m0.z * v.z;
406     result.y = m1.x * v.x + m1.y * v.y + m1.z * v.z;
407     result.z = m2.x * v.x + m2.y * v.y + m2.z * v.z;
408     return result;
409 }
410 
411 // Multiplies the transpose of the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the
normal n.
412 // Note that the given matrix is supposed to be the inverse of the actual transformation matrix.
413 static __forceinline__ __device__ float3 optixTransformNormal(const float4& m0, const float4& m1, const
float4& m2, const float3& n)
414 {
415     float3 result;
416     result.x = m0.x * n.x + m1.x * n.y + m2.x * n.z;
417     result.y = m0.y * n.x + m1.y * n.y + m2.y * n.z;
418     result.z = m0.z * n.x + m1.z * n.y + m2.z * n.z;
419     return result;
420 }
421 
422 }  // namespace optix_impl
423 
424 #endif
```

8.7 optix.h File Reference

Macros

- #define OPTIX_VERSION 70600

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 70600
```

The OptiX version.

- major = OPTIX_VERSION/10000
- minor = (OPTIX_VERSION%10000)/100
- micro = OPTIX_VERSION%100

8.8 optix.h

Go to the documentation of this file.

```
1
2 /*
3 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
4 *
5 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
6 * rights in and to this software, related documentation and any modifications thereto.
7 * Any use, reproduction, disclosure or distribution of this software and related
8 * documentation without an express license agreement from NVIDIA Corporation is strictly
9 * prohibited.
10 *
11 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
12 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
13 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
14 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
15 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
16 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
17 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
18 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
19 * SUCH DAMAGES
20 */
21
28
29 #ifndef __optix_optix_h__
30 #define __optix_optix_h__
31
37 #define OPTIX_VERSION 70600
38
39
40 #ifdef __CUDACC__
41 #include "optix_device.h"
42 #else
43 #include "optix_host.h"
44 #endif
45
46
47 #endif  // __optix_optix_h__
```

8.9 optix_7_device.h File Reference

Functions

- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_5 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_6 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_7 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_8 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_9 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_10 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_11 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_12 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_13 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_14 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_15 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_16 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_17 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_18 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_19 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_20 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_21 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_22 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_23 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_24 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_25 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_26 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_27 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_28 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_29 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_30 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_31 (unsigned int p)
- static __forceinline__ __device__ unsigned int optixGetPayload_0 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_1 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_2 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_3 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_4 ()

- static __forceinline__ __device__ unsigned int optixGetPayload_5 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_6 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_7 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_8 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_9 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_10 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_11 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_12 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_13 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_14 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_15 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_16 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_17 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_18 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_19 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_20 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_21 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_22 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_23 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_24 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_25 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_26 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_27 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_28 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_29 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_30 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_31 ()
- static __forceinline__ __device__ void optixSetPayloadTypes (unsigned int typeMask)
- static __forceinline__ __device__ unsigned int optixUndefinedValue ()
- static __forceinline__ __device__ float3 optixGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixGetObjectRayOrigin ()
- static __forceinline__ __device__ float3 optixGetObjectRayDirection ()
- static __forceinline__ __device__ float optixGetRayTmin ()
- static __forceinline__ __device__ float optixGetRayTmax ()
- static __forceinline__ __device__ float optixGetRayTime ()
- static __forceinline__ __device__ unsigned int optixGetRayFlags ()
- static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceTraversableFromIAS (OptixTraversableHandle ias, unsigned int instIdx)
- static __forceinline__ __device__ void optixGetTriangleVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static __forceinline__ __device__ void optixGetLinearCurveVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetCubicBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])

- static __forceinline__ __device__ void optixGetSphereData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle ()
- static __forceinline__ __device__ float optixGetGASMotionTimeBegin (OptixTraversableHandle gas)
- static __forceinline__ __device__ float optixGetGASMotionTimeEnd (OptixTraversableHandle gas)
- static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (OptixTraversableHandle gas)
- static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (float m[12])
- static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (float3 normal)
- static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (float3 normal)
- static __forceinline__ __device__ unsigned int optixGetTransformListSize ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTransformType optixGetTransformTypeFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixStaticTransform * optixGetStaticTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixSRTMotionTransform * optixGetSRTMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixMatrixMotionTransform * optixGetMatrixMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceChildFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceInverseTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3)

- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6, unsigned int a7)
- static __forceinline__ __device__ unsigned int optixGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_7 ()
- static __forceinline__ __device__ void optixTerminateRay ()
- static __forceinline__ __device__ void optixIgnoreIntersection ()
- static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixGetHitKind ()
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsFrontFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsBackFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ()
- static __forceinline__ __device__ bool optixIsFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsBackFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleHit ()
- static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ()
- static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ()
- static __forceinline__ __device__ float optixGetCurveParameter ()
- static __forceinline__ __device__ uint3 optixGetLaunchIndex ()
- static __forceinline__ __device__ uint3 optixGetLaunchDimensions ()
- static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ()
- static __forceinline__ __device__ void optixThrowException (int exceptionCode)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)

- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6, unsigned int exceptionDetail7)
- static __forceinline__ __device__ int optixGetExceptionCode ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7 ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable ()
- static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset ()
- static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay ()
- static __forceinline__ __device__ OptixParameterMismatchExceptionDetails optixGetExceptionParameterMismatch ()
- static __forceinline__ __device__ char * optixGetExceptionLineInfo ()
- template<typename ReturnT , typename... ArgTypes> static __forceinline__ __device__ ReturnT optixDirectCall (unsigned int sbtIndex, ArgTypes... args)
- template<typename ReturnT , typename... ArgTypes> static __forceinline__ __device__ ReturnT optixContinuationCall (unsigned int sbtIndex, ArgTypes... args)
- static __forceinline__ __device__ uint4 optixTexFootprint2D (unsigned long long tex, unsigned int texInfo, float x, float y, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DLod (unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool coarse, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DGrad (unsigned long long tex, unsigned int texInfo, float x, float y, float dPdx_x, float dPdx_y, float dPdy_x, float dPdy_y, bool coarse, unsigned int *singleMipLevel)

8.9.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

OptiX public API Reference - Device API declarations

8.10 optix_7_device.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
3 *
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly
8 * prohibited.
9 *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
26
27 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
28 #error("optix_7_device.h is an internal header file and must not be used directly.  Please use
optix_device.h or optix.h instead.")
29 #endif
30
31
32 #ifndef __optix_optix_7_device_h__
33 #define __optix_optix_7_device_h__
34
35 #if defined(__cplusplus) && (__cplusplus < 201103L) && !defined(_WIN32)
36 #error Device code for OptiX requires at least C++11. Consider adding "--std c++11" to the nvcc
command-line.
37 #endif
38
39 #include "optix_7_types.h"
40
43
62 template <typename... Payload>
63 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
64                                                   float3                rayOrigin,
65                                                   float3                rayDirection,
66                                                   float                 tmin,
67                                                   float                 tmax,
68                                                   float                 rayTime,
69                                                   OptixVisibilityMask   visibilityMask,
70                                                   unsigned int          rayFlags,
71                                                   unsigned int          SBToffset,
72                                                   unsigned int          SBTstride,
73                                                   unsigned int          missSBTIndex,
74                                                   Payload&...             payload);
75
91 template <typename... Payload>
92 static __forceinline__ __device__ void optixTrace(OptixPayloadTypeID     type,
93                                                   OptixTraversableHandle handle,
94                                                   float3                rayOrigin,
95                                                   float3                rayDirection,
96                                                   float                 tmin,
97                                                   float                 tmax,
98                                                   float                 rayTime,
99                                                   OptixVisibilityMask   visibilityMask,
100                                                   unsigned int          rayFlags,
101                                                   unsigned int          SBToffset,
102                                                   unsigned int          SBTstride,
```

```
103                                                 unsigned int           missSBTIndex,
104                                                 Payload&...              payload);
105
107 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p);
109 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p);
111 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p);
113 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p);
115 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p);
117 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p);
119 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p);
121 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p);
122
124 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p);
126 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p);
128 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p);
130 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p);
132 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p);
134 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p);
136 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p);
138 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p);
140 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p);
142 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p);
144 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p);
146 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p);
148 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p);
150 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p);
152 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p);
154 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p);
156 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p);
158 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p);
160 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p);
162 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p);
164 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p);
166 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p);
168 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p);
170 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p);
171
173 static __forceinline__ __device__ unsigned int optixGetPayload_0();
175 static __forceinline__ __device__ unsigned int optixGetPayload_1();
177 static __forceinline__ __device__ unsigned int optixGetPayload_2();
179 static __forceinline__ __device__ unsigned int optixGetPayload_3();
181 static __forceinline__ __device__ unsigned int optixGetPayload_4();
183 static __forceinline__ __device__ unsigned int optixGetPayload_5();
185 static __forceinline__ __device__ unsigned int optixGetPayload_6();
187 static __forceinline__ __device__ unsigned int optixGetPayload_7();
188
190 static __forceinline__ __device__ unsigned int optixGetPayload_8();
192 static __forceinline__ __device__ unsigned int optixGetPayload_9();
194 static __forceinline__ __device__ unsigned int optixGetPayload_10();
196 static __forceinline__ __device__ unsigned int optixGetPayload_11();
198 static __forceinline__ __device__ unsigned int optixGetPayload_12();
200 static __forceinline__ __device__ unsigned int optixGetPayload_13();
202 static __forceinline__ __device__ unsigned int optixGetPayload_14();
204 static __forceinline__ __device__ unsigned int optixGetPayload_15();
206 static __forceinline__ __device__ unsigned int optixGetPayload_16();
208 static __forceinline__ __device__ unsigned int optixGetPayload_17();
210 static __forceinline__ __device__ unsigned int optixGetPayload_18();
212 static __forceinline__ __device__ unsigned int optixGetPayload_19();
214 static __forceinline__ __device__ unsigned int optixGetPayload_20();
216 static __forceinline__ __device__ unsigned int optixGetPayload_21();
218 static __forceinline__ __device__ unsigned int optixGetPayload_22();
220 static __forceinline__ __device__ unsigned int optixGetPayload_23();
222 static __forceinline__ __device__ unsigned int optixGetPayload_24();
224 static __forceinline__ __device__ unsigned int optixGetPayload_25();
226 static __forceinline__ __device__ unsigned int optixGetPayload_26();
228 static __forceinline__ __device__ unsigned int optixGetPayload_27();
230 static __forceinline__ __device__ unsigned int optixGetPayload_28();
```

```
232 static __forceinline__ __device__ unsigned int optixGetPayload_29();
234 static __forceinline__ __device__ unsigned int optixGetPayload_30();
236 static __forceinline__ __device__ unsigned int optixGetPayload_31();
237
244 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int typeMask);
245
247 static __forceinline__ __device__ unsigned int optixUndefinedValue();
248
254 static __forceinline__ __device__ float3 optixGetWorldRayOrigin();
255
261 static __forceinline__ __device__ float3 optixGetWorldRayDirection();
262
266 static __forceinline__ __device__ float3 optixGetObjectRayOrigin();
267
271 static __forceinline__ __device__ float3 optixGetObjectRayDirection();
272
276 static __forceinline__ __device__ float optixGetRayTmin();
277
282 static __forceinline__ __device__ float optixGetRayTmax();
283
288 static __forceinline__ __device__ float optixGetRayTime();
289
293 static __forceinline__ __device__ unsigned int optixGetRayFlags();
294
298 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask();
299
302 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias, unsigned int instIdx);
303
310 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas, unsigned
int primIdx, unsigned int sbtGASIndex, float time, float3 data[3]);
311
312
320 static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2]);
321
329 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3]);
330
338 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4]);
339
347 static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas, unsigned
int primIdx, unsigned int sbtGASIndex, float time, float4 data[4]);
348
355 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas, unsigned int
primIdx, unsigned int sbtGASIndex, float time, float4 data[1]);
356
359 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle();
360
362 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle gas);
363
365 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle gas);
366
368 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle gas);
369
373 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12]);
374
378 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12]);
379
384 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point);
385
390 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec);
391
396 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal);
397
402 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point);
```

```
403
408 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec);
409
414 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal);
415
419 static __forceinline__ __device__ unsigned int optixGetTransformListSize();
420
424 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int index);
425
426
428 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle);
429
433 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle);
434
438 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle);
439
443 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle);
444
448 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle);
449
453 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle(OptixTraversableHandle handle);
454
458 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle);
459
463 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle);
464
481 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind);
482
486 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0);
487
491 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1);
492
496 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1, unsigned int a2);
497
501 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
502                                                                unsigned int hitKind,
503                                                                unsigned int a0,
504                                                                unsigned int a1,
505                                                                unsigned int a2,
506                                                                unsigned int a3);
507
511 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
512                                                                unsigned int hitKind,
513                                                                unsigned int a0,
514                                                                unsigned int a1,
515                                                                unsigned int a2,
516                                                                unsigned int a3,
517                                                                unsigned int a4);
518
522 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
523                                                                unsigned int hitKind,
524                                                                unsigned int a0,
525                                                                unsigned int a1,
526                                                                unsigned int a2,
527                                                                unsigned int a3,
528                                                                unsigned int a4,
```

```
529                                                                 unsigned int a5);
530
534 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
535                                                                 unsigned int hitKind,
536                                                                 unsigned int a0,
537                                                                 unsigned int a1,
538                                                                 unsigned int a2,
539                                                                 unsigned int a3,
540                                                                 unsigned int a4,
541                                                                 unsigned int a5,
542                                                                 unsigned int a6);
543
547 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
548                                                                 unsigned int hitKind,
549                                                                 unsigned int a0,
550                                                                 unsigned int a1,
551                                                                 unsigned int a2,
552                                                                 unsigned int a3,
553                                                                 unsigned int a4,
554                                                                 unsigned int a5,
555                                                                 unsigned int a6,
556                                                                 unsigned int a7);
557
559 static __forceinline__ __device__ unsigned int optixGetAttribute_0();
561 static __forceinline__ __device__ unsigned int optixGetAttribute_1();
563 static __forceinline__ __device__ unsigned int optixGetAttribute_2();
565 static __forceinline__ __device__ unsigned int optixGetAttribute_3();
567 static __forceinline__ __device__ unsigned int optixGetAttribute_4();
569 static __forceinline__ __device__ unsigned int optixGetAttribute_5();
571 static __forceinline__ __device__ unsigned int optixGetAttribute_6();
573 static __forceinline__ __device__ unsigned int optixGetAttribute_7();
574
578 static __forceinline__ __device__ void optixTerminateRay();
579
583 static __forceinline__ __device__ void optixIgnoreIntersection();
584
585
597 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex();
598
604 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex();
605
606
614 static __forceinline__ __device__ unsigned int optixGetInstanceId();
615
621 static __forceinline__ __device__ unsigned int optixGetInstanceIndex();
622
631 static __forceinline__ __device__ unsigned int optixGetHitKind();
632
634 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind);
635
637 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind);
638
640 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind);
641
643 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType();
644
646 static __forceinline__ __device__ bool optixIsFrontFaceHit();
647
649 static __forceinline__ __device__ bool optixIsBackFaceHit();
650
652 static __forceinline__ __device__ bool optixIsTriangleHit();
653
655 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit();
656
658 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit();
659
664 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics();
```

```
665
670 static __forceinline__ __device__ float optixGetCurveParameter();
671
675 static __forceinline__ __device__ uint3 optixGetLaunchIndex();
676
678 static __forceinline__ __device__ uint3 optixGetLaunchDimensions();
679
681 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer();
682
694 static __forceinline__ __device__ void optixThrowException(int exceptionCode);
695
699 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0);
700
704 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
705                                                            unsigned int exceptionDetail0,
706                                                            unsigned int exceptionDetail1);
707
711 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
712                                                            unsigned int exceptionDetail0,
713                                                            unsigned int exceptionDetail1,
714                                                            unsigned int exceptionDetail2);
715
719 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
720                                                            unsigned int exceptionDetail0,
721                                                            unsigned int exceptionDetail1,
722                                                            unsigned int exceptionDetail2,
723                                                            unsigned int exceptionDetail3);
724
728 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
729                                                            unsigned int exceptionDetail0,
730                                                            unsigned int exceptionDetail1,
731                                                            unsigned int exceptionDetail2,
732                                                            unsigned int exceptionDetail3,
733                                                            unsigned int exceptionDetail4);
734
738 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
739                                                            unsigned int exceptionDetail0,
740                                                            unsigned int exceptionDetail1,
741                                                            unsigned int exceptionDetail2,
742                                                            unsigned int exceptionDetail3,
743                                                            unsigned int exceptionDetail4,
744                                                            unsigned int exceptionDetail5);
745
749 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
750                                                            unsigned int exceptionDetail0,
751                                                            unsigned int exceptionDetail1,
752                                                            unsigned int exceptionDetail2,
753                                                            unsigned int exceptionDetail3,
754                                                            unsigned int exceptionDetail4,
755                                                            unsigned int exceptionDetail5,
756                                                            unsigned int exceptionDetail6);
757
761 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
762                                                            unsigned int exceptionDetail0,
763                                                            unsigned int exceptionDetail1,
764                                                            unsigned int exceptionDetail2,
765                                                            unsigned int exceptionDetail3,
766                                                            unsigned int exceptionDetail4,
767                                                            unsigned int exceptionDetail5,
768                                                            unsigned int exceptionDetail6,
769                                                            unsigned int exceptionDetail7);
770
774 static __forceinline__ __device__ int optixGetExceptionCode();
775
782 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0();
783
```

```
787 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1();
788
792 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2();
793
797 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3();
798
802 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4();
803
807 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5();
808
812 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6();
813
817 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7();
818
824 static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable();
825
831 static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset();
832
841 static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay();
842
855 static __forceinline__ __device__ OptixParameterMismatchExceptionDetails
optixGetExceptionParameterMismatch();
856
867 static __forceinline__ __device__ char* optixGetExceptionLineInfo();
868
883 template <typename ReturnT, typename... ArgTypes>
884 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args);
885
886
902 template <typename ReturnT, typename... ArgTypes>
903 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes... args);
904
905
968 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int texInfo,
float x, float y, unsigned int* singleMipLevel);
969
979 static __forceinline__ __device__ uint4
980 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool
coarse, unsigned int* singleMipLevel);
981
994 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
995                                                                 unsigned int       texInfo,
996                                                                 float              x,
997                                                                 float              y,
998                                                                 float              dPdx_x,
999                                                                 float              dPdx_y,
1000                                                                 float              dPdy_x,
1001                                                                 float              dPdy_y,
1002                                                                 bool               coarse,
1003                                                                 unsigned int*      singleMipLevel);
1004   // end group optix_device_api
1006
1007 #include "internal/optix_7_device_impl.h"
1008
1009 #endif  // __optix_optix_7_device_h__
```

8.11 optix_7_host.h File Reference

Functions

- const char * optixGetErrorName (OptixResult result)
- const char * optixGetErrorString (OptixResult result)
- OptixResult optixDeviceContextCreate (CUcontext fromContext, const OptixDeviceContextOptions *options, OptixDeviceContext *context)
- OptixResult optixDeviceContextDestroy (OptixDeviceContext context)

- OptixResult optixDeviceContextGetProperty (OptixDeviceContext context, OptixDeviceProperty property, void *value, size_t sizeInBytes)
- OptixResult optixDeviceContextSetLogCallback (OptixDeviceContext context, OptixLogCallback callbackFunction, void *callbackData, unsigned int callbackLevel)
- OptixResult optixDeviceContextSetCacheEnabled (OptixDeviceContext context, int enabled)
- OptixResult optixDeviceContextSetCacheLocation (OptixDeviceContext context, const char *location)
- OptixResult optixDeviceContextSetCacheDatabaseSizes (OptixDeviceContext context, size_t lowWaterMark, size_t highWaterMark)
- OptixResult optixDeviceContextGetCacheEnabled (OptixDeviceContext context, int *enabled)
- OptixResult optixDeviceContextGetCacheLocation (OptixDeviceContext context, char *location, size_t locationSize)
- OptixResult optixDeviceContextGetCacheDatabaseSizes (OptixDeviceContext context, size_t *lowWaterMark, size_t *highWaterMark)
- OptixResult optixPipelineCreate (OptixDeviceContext context, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixPipelineLinkOptions *pipelineLinkOptions, const OptixProgramGroup *programGroups, unsigned int numProgramGroups, char *logString, size_t *logStringSize, OptixPipeline *pipeline)
- OptixResult optixPipelineDestroy (OptixPipeline pipeline)
- OptixResult optixPipelineSetStackSize (OptixPipeline pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)
- OptixResult optixModuleCreateFromPTX (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *PTX, size_t PTXsize, char *logString, size_t *logStringSize, OptixModule *module)
- OptixResult optixModuleCreateFromPTXWithTasks (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *PTX, size_t PTXsize, char *logString, size_t *logStringSize, OptixModule *module, OptixTask *firstTask)
- OptixResult optixModuleGetCompilationState (OptixModule module, OptixModuleCompileState *state)
- OptixResult optixModuleDestroy (OptixModule module)
- OptixResult optixBuiltinISModuleGet (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixBuiltinISOptions *builtinISOptions, OptixModule *builtinModule)
- OptixResult optixTaskExecute (OptixTask task, OptixTask *additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int *numAdditionalTasksCreated)
- OptixResult optixProgramGroupGetStackSize (OptixProgramGroup programGroup, OptixStackSizes *stackSizes)
- OptixResult optixProgramGroupCreate (OptixDeviceContext context, const OptixProgramGroupDesc *programDescriptions, unsigned int numProgramGroups, const OptixProgramGroupOptions *options, char *logString, size_t *logStringSize, OptixProgramGroup *programGroups)
- OptixResult optixProgramGroupDestroy (OptixProgramGroup programGroup)
- OptixResult optixLaunch (OptixPipeline pipeline, CUstream stream, CUdeviceptr pipelineParams, size_t pipelineParamsSize, const OptixShaderBindingTable *sbt, unsigned int width, unsigned int height, unsigned int depth)
- OptixResult optixSbtRecordPackHeader (OptixProgramGroup programGroup, void *sbtRecordHeaderHostPointer)

- OptixResult optixAccelComputeMemoryUsage (OptixDeviceContext context, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, OptixAccelBufferSizes *bufferSizes)
- OptixResult optixAccelBuild (OptixDeviceContext context, CUstream stream, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, CUdeviceptr tempBuffer, size_t tempBufferSizeInBytes, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle, const OptixAccelEmitDesc *emittedProperties, unsigned int numEmittedProperties)
- OptixResult optixAccelGetRelocationInfo (OptixDeviceContext context, OptixTraversableHandle handle, OptixRelocationInfo *info)
- OptixResult optixCheckRelocationCompatibility (OptixDeviceContext context, const OptixRelocationInfo *info, int *compatible)
- OptixResult optixAccelRelocate (OptixDeviceContext context, CUstream stream, const OptixRelocationInfo *info, const OptixRelocateInput *relocateInputs, size_t numRelocateInputs, CUdeviceptr targetAccel, size_t targetAccelSizeInBytes, OptixTraversableHandle *targetHandle)
- OptixResult optixAccelCompact (OptixDeviceContext context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle)
- OptixResult 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 optixDenoiserCreate (OptixDeviceContext context, OptixDenoiserModelKind modelKind, const OptixDenoiserOptions *options, OptixDenoiser *denoiser)
- OptixResult optixDenoiserCreateWithUserModel (OptixDeviceContext context, const void *userData, size_t userDataSizeInBytes, OptixDenoiser *denoiser)
- OptixResult optixDenoiserDestroy (OptixDenoiser denoiser)
- OptixResult optixDenoiserComputeMemoryResources (const OptixDenoiser denoiser, unsigned int outputWidth, unsigned int outputHeight, OptixDenoiserSizes *returnSizes)
- OptixResult optixDenoiserSetup (OptixDenoiser denoiser, CUstream stream, unsigned int inputWidth, unsigned int inputHeight, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OptixResult optixDenoiserInvoke (OptixDenoiser denoiser, CUstream stream, const OptixDenoiserParams *params, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, const OptixDenoiserGuideLayer *guideLayer, const OptixDenoiserLayer *layers, unsigned int numLayers, unsigned int inputOffsetX, unsigned int inputOffsetY, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OptixResult optixDenoiserComputeIntensity (OptixDenoiser denoiser, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputIntensity, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OptixResult optixDenoiserComputeAverageColor (OptixDenoiser denoiser, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputAverageColor, CUdeviceptr scratch, size_t scratchSizeInBytes)

8.11.1 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

8.11.2 Function Documentation

8.11.2.1 optixAccelBuild()

```
OptixResult optixAccelBuild (
            OptixDeviceContext context,
            CUstream stream,
            const OptixAccelBuildOptions * accelOptions,
            const OptixBuildInput * buildInputs,
            unsigned int numBuildInputs,
            CUdeviceptr tempBuffer,
            size_t tempBufferSizeInBytes,
            CUdeviceptr outputBuffer,
            size_t outputBufferSizeInBytes,
            OptixTraversableHandle * outputHandle,
            const OptixAccelEmitDesc * emittedProperties,
            unsigned int numEmittedProperties )
```

Parameters

in	*context*	
in	*stream*	
in	*accelOptions*	accel options
in	*buildInputs*	an array of OptixBuildInput objects
in	*numBuildInputs*	must be >= 1 for GAS, and == 1 for IAS
in	*tempBuffer*	must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT
in	*tempBufferSizeInBytes*	
in	*outputBuffer*	must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT
in	*outputBufferSizeInBytes*	
out	*outputHandle*	
in	*emittedProperties*	types of requested properties and output buffers
in	*numEmittedProperties*	number of post-build properties to populate (may be zero)

8.11.2.2 optixAccelCompact()

```
OptixResult optixAccelCompact (
          OptixDeviceContext context,
          CUstream stream,
          OptixTraversableHandle inputHandle,
          CUdeviceptr outputBuffer,
          size_t outputBufferSizeInBytes,
          OptixTraversableHandle * outputHandle )
```

After building an acceleration structure, it can be copied in a compacted form to reduce memory. In order to be compacted, OPTIX_BUILD_FLAG_ALLOW_COMPACTION must be supplied in OptixAccelBuildOptions::buildFlags passed to optixAccelBuild.

'outputBuffer' is the pointer to where the compacted acceleration structure will be written. This pointer must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT.

The size of the memory specified in 'outputBufferSizeInBytes' should be at least the value computed using the OPTIX_PROPERTY_TYPE_COMPACTED_SIZE that was reported during optixAccelBuild.

Parameters

in	*context*
in	*stream*
in	*inputHandle*
in	*outputBuffer*
in	*outputBufferSizeInBytes*
out	*outputHandle*

8.11.2.3 optixAccelComputeMemoryUsage()

```
OptixResult optixAccelComputeMemoryUsage (
          OptixDeviceContext context,
          const OptixAccelBuildOptions * accelOptions,
          const OptixBuildInput * buildInputs,
          unsigned int numBuildInputs,
          OptixAccelBufferSizes * bufferSizes )
```

Parameters

in	*context*	
in	*accelOptions*	options for the accel build
in	*buildInputs*	an array of OptixBuildInput objects
in	*numBuildInputs*	number of elements in buildInputs (must be at least 1)
out	*bufferSizes*	fills in buffer sizes

8.11.2.4 optixAccelGetRelocationInfo()

```
OptixResult optixAccelGetRelocationInfo (
            OptixDeviceContext context,
            OptixTraversableHandle handle,
            OptixRelocationInfo * info )
```

Obtain relocation information, stored in OptixRelocationInfo, for a given context and acceleration structure's traversable handle.

The relocation information can be passed to optixCheckRelocationCompatibility to determine if an acceleration structure, referenced by 'handle', can be relocated to a different device's memory space (see optixCheckRelocationCompatibility).

When used with optixAccelRelocate, it provides data necessary for doing the relocation.

If the acceleration structure data associated with 'handle' is copied multiple times, the same OptixRelocationInfo can also be used on all copies.

Parameters

in	*context*
in	*handle*
out	*info*

Returns

OPTIX_ERROR_INVALID_VALUE will be returned for traversable handles that are not from acceleration structure builds.

8.11.2.5 optixAccelRelocate()

```
OptixResult optixAccelRelocate (
            OptixDeviceContext context,
            CUstream stream,
            const OptixRelocationInfo * info,
            const OptixRelocateInput * relocateInputs,
            size_t numRelocateInputs,
            CUdeviceptr targetAccel,
            size_t targetAccelSizeInBytes,
            OptixTraversableHandle * targetHandle )
```

optixAccelRelocate is called to update the acceleration structure after it has been relocated. Relocation is necessary when the acceleration structure's location in device memory has changed. optixAccelRelocate does not copy the memory. This function only operates on the relocated memory whose new location is specified by 'targetAccel'. optixAccelRelocate also returns the new OptixTraversableHandle associated with 'targetAccel'. The original memory (source) is not required to be valid, only the OptixRelocationInfo.

Before calling optixAccelRelocate, optixCheckRelocationCompatibility should be called to ensure the copy will be compatible with the destination device context.

The memory pointed to by 'targetAccel' should be allocated with the same size as the source acceleration. Similar to the 'outputBuffer' used in optixAccelBuild, this pointer must be a multiple of

OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT.

The memory in 'targetAccel' must be allocated as long as the accel is in use.

The instance traversables referenced by an IAS and the micromaps referenced by a triangle GAS may themselves require relocation. 'relocateInputs' and 'numRelocateInputs' should be used to specify the relocated traversables and micromaps. After relocation, the relocated accel will reference these relocated traversables and micromaps instead of their sources. The number of relocate inputs 'numRelocateInputs' must match the number of build inputs 'numBuildInputs' used to build the source accel. Relocation inputs correspond with build inputs used to build the source accel and should appear in the same order (see optixAccelBuild). 'relocateInputs' and 'numRelocateInputs' may be zero, preserving any references to traversables and micromaps from the source accel.

Parameters

in	*context*
in	*stream*
in	*info*
in	*relocateInputs*
in	*numRelocateInputs*
in	*targetAccel*
in	*targetAccelSizeInBytes*
out	*targetHandle*

8.11.2.6 optixBuiltinISModuleGet()

```
OptixResult optixBuiltinISModuleGet (
            OptixDeviceContext context,
            const OptixModuleCompileOptions * moduleCompileOptions,
            const OptixPipelineCompileOptions * pipelineCompileOptions,
            const OptixBuiltinISOptions * builtinISOptions,
            OptixModule * builtinModule )
```

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

8.11.2.7 optixCheckRelocationCompatibility()

```
OptixResult optixCheckRelocationCompatibility (
            OptixDeviceContext context,
            const OptixRelocationInfo * info,
            int * compatible )
```

Checks if an optix data structure built using another OptixDeviceContext (that was used to fill in 'info') is compatible with the OptixDeviceContext specified in the 'context' parameter.

Any device is always compatible with itself.

Parameters

Parameters

in	*context*	
in	*info*	
out	*compatible*	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.

8.11.2.8 optixConvertPointerToTraversableHandle()

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

Parameters

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

8.11.2.9 optixDenoiserComputeAverageColor()

```
OptixResult optixDenoiserComputeAverageColor (
            OptixDenoiser denoiser,
            CUstream stream,
            const OptixImage2D * inputImage,
            CUdeviceptr outputAverageColor,
            CUdeviceptr scratch,
            size_t scratchSizeInBytes )
```

Compute average logarithmic for each of the first three channels for the given image. When denoising tiles the intensity of the entire image should be computed, i.e. not per tile to get consistent results.

The size of scratch memory required can be queried with optixDenoiserComputeMemoryResources.

data type unsigned char is not supported for 'inputImage', it must be 3 or 4 component half/float.

Parameters

in	*denoiser*	
in	*stream*	

Parameters

in	*inputImage*	
out	*outputAverageColor*	three floats
in	*scratch*	
in	*scratchSizeInBytes*	

8.11.2.10 optixDenoiserComputeIntensity()

```
OptixResult optixDenoiserComputeIntensity (
            OptixDenoiser denoiser,
            CUstream stream,
            const OptixImage2D * inputImage,
            CUdeviceptr outputIntensity,
            CUdeviceptr scratch,
            size_t scratchSizeInBytes )
```

Computes the logarithmic average intensity of the given image. The returned value 'outputIntensity' is multiplied with the RGB values of the input image/tile in optixDenoiserInvoke if given in the parameter OptixDenoiserParams::hdrIntensity (otherwise 'hdrIntensity' must be a null pointer). This is useful for denoising HDR images which are very dark or bright. When denoising tiles the intensity of the entire image should be computed, i.e. not per tile to get consistent results.

For each RGB pixel in the inputImage the intensity is calculated and summed if it is greater than 1e-8f: intensity = log(r * 0.212586f + g * 0.715170f + b * 0.072200f). The function returns 0.18 / exp(sum of intensities / 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	*denoiser*	
in	*stream*	
in	*inputImage*	
out	*outputIntensity*	single float
in	*scratch*	
in	*scratchSizeInBytes*	

8.11.2.11 optixDenoiserComputeMemoryResources()

```
OptixResult optixDenoiserComputeMemoryResources (
            const OptixDenoiser denoiser,
            unsigned int outputWidth,
            unsigned int outputHeight,
            OptixDenoiserSizes * returnSizes )
```

Computes the GPU memory resources required to execute the denoiser.

Memory for state and scratch buffers must be allocated with the sizes in 'returnSizes' and scratch memory passed to optixDenoiserSetup, optixDenoiserInvoke, optixDenoiserComputeIntensity and optixDenoiserComputeAverageColor. For tiled denoising an overlap area ('overlapWindowSizeInPixels') must be added to each tile on all sides which increases the amount of memory needed to denoise a tile. In case of tiling use withOverlapScratchSizeInBytes for scratch memory size. If only full resolution images are denoised, withoutOverlapScratchSizeInBytes can be used which is always smaller than withOverlapScratchSizeInBytes.

'outputWidth' and 'outputHeight' is the dimension of the image to be denoised (without overlap in case tiling is being used). 'outputWidth' and 'outputHeight' must be greater than or equal to the dimensions passed to optixDenoiserSetup.

Parameters

in	*denoiser*
in	*outputWidth*
in	*outputHeight*
out	*returnSizes*

8.11.2.12 optixDenoiserCreate()

```
OptixResult optixDenoiserCreate (
            OptixDeviceContext context,
            OptixDenoiserModelKind modelKind,
            const OptixDenoiserOptions * options,
            OptixDenoiser * denoiser )
```

Creates a denoiser object with the given options, using built-in inference models.

'modelKind' selects the model used for inference. Inference for the built-in models can be guided (giving hints to improve image quality) with albedo and normal vector images in the guide layer (see 'optixDenoiserInvoke'). Use of these images must be enabled in 'OptixDenoiserOptions'.

Parameters

in	*context*
in	*modelKind*
in	*options*
out	*denoiser*

8.11.2.13 optixDenoiserCreateWithUserModel()

```
OptixResult optixDenoiserCreateWithUserModel (
            OptixDeviceContext context,
            const void * userData,
            size_t userDataSizeInBytes,
            OptixDenoiser * denoiser )
```

Creates a denoiser object with the given options, using a provided inference model.

'userData' and 'userDataSizeInBytes' provide a user model for inference. The memory passed in userData will be accessed only during the invocation of this function and can be freed after it returns. The user model must export only one weight set which determines both the model kind and the required set of guide images.

Parameters

in	*context*
in	*userData*
in	*userDataSizeInBytes*
out	*denoiser*

8.11.2.14 optixDenoiserDestroy()

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

Destroys the denoiser object and any associated host resources.

8.11.2.15 optixDenoiserInvoke()

```
OptixResult optixDenoiserInvoke (
          OptixDenoiser denoiser,
          CUstream stream,
          const OptixDenoiserParams * params,
          CUdeviceptr denoiserState,
          size_t denoiserStateSizeInBytes,
          const OptixDenoiserGuideLayer * guideLayer,
          const OptixDenoiserLayer * layers,
          unsigned int numLayers,
          unsigned int inputOffsetX,
          unsigned int inputOffsetY,
          CUdeviceptr scratch,
          size_t scratchSizeInBytes )
```

Invokes denoiser on a set of input data and produces at least one output image. State memory must be available during the execution of the denoiser (or until optixDenoiserSetup is called with a new state memory pointer). Scratch memory passed is used only for the duration of this function. Scratch and state memory sizes must have a size greater than or equal to the sizes as returned by optixDenoiserComputeMemoryResources.

'inputOffsetX' and 'inputOffsetY' are pixel offsets in the 'inputLayers' image specifying the beginning of the image without overlap. When denoising an entire image without tiling there is no overlap and 'inputOffsetX' and 'inputOffsetY' must be zero. When denoising a tile which is adjacent to one of the four sides of the entire image the corresponding offsets must also be zero since there is no overlap at the side adjacent to the image border.

'guideLayer' provides additional information to the denoiser. When providing albedo and normal vector guide images, the corresponding fields in the 'OptixDenoiserOptions' must be enabled, see optixDenoiserCreate. 'guideLayer' must not be null. If a guide image in 'OptixDenoiserOptions' is not enabled, the corresponding image in 'OptixDenoiserGuideLayer' is ignored.

If OPTIX_DENOISER_MODEL_KIND_TEMPORAL or OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV is selected, a 2d flow image must be given in 'OptixDenoiserGuideLayer'. It describes for each pixel the flow from the previous to the current frame (a 2d vector in pixel space). The denoised beauty/AOV of the previous frame must be given in 'previousOutput'. If this image is not available in the first frame of a sequence, the noisy beauty/AOV from the first frame and zero flow vectors could be given as a substitute. For non-temporal model kinds the flow image in 'OptixDenoiserGuideLayer' is ignored. 'previousOutput' and 'output' may refer to the same buffer, 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, 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.

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	*denoiser*
in	*stream*
in	*params*
in	*denoiserState*
in	*denoiserStateSizeInBytes*
in	*guideLayer*
in	*layers*
in	*numLayers*
in	*inputOffsetX*
in	*inputOffsetY*
in	*scratch*
in	*scratchSizeInBytes*

8.11.2.16 optixDenoiserSetup()

```
OptixResult optixDenoiserSetup (
            OptixDenoiser denoiser,
            CUstream stream,
            unsigned int inputWidth,
```

```
    unsigned int inputHeight,
    CUdeviceptr denoiserState,
    size_t denoiserStateSizeInBytes,
    CUdeviceptr scratch,
    size_t scratchSizeInBytes )
```

Initializes the state required by the denoiser.

'inputWidth' and 'inputHeight' must include overlap on both sides of the image if tiling is being used. The overlap is returned by optixDenoiserComputeMemoryResources. For subsequent calls to optixDenoiserInvoke 'inputWidth' and 'inputHeight' are the maximum dimensions of the input layers. Dimensions of the input layers passed to optixDenoiserInvoke may be different in each invocation however they always must be smaller than 'inputWidth' and 'inputHeight' passed to optixDenoiserSetup.

Parameters

in	*denoiser*
in	*stream*
in	*inputWidth*
in	*inputHeight*
in	*denoiserState*
in	*denoiserStateSizeInBytes*
in	*scratch*
in	*scratchSizeInBytes*

8.11.2.17 optixDeviceContextCreate()

```
OptixResult optixDeviceContextCreate (
    CUcontext fromContext,
    const OptixDeviceContextOptions * options,
    OptixDeviceContext * context )
```

Create a device context associated with the CUDA context specified with 'fromContext'.

If zero is specified for 'fromContext', OptiX will use the current CUDA context. The CUDA context should be initialized before calling optixDeviceContextCreate.

Parameters

in	*fromContext*
in	*options*
out	*context*

Returns

- OPTIX_ERROR_CUDA_NOT_INITIALIZED If using zero for 'fromContext' and CUDA has not been initialized yet on the calling thread.
- OPTIX_ERROR_CUDA_ERROR CUDA operation failed.
- OPTIX_ERROR_HOST_OUT_OF_MEMORY Heap allocation failed.
- OPTIX_ERROR_INTERNAL_ERROR Internal error

8.11.2.18 optixDeviceContextDestroy()

```
OptixResult optixDeviceContextDestroy (
          OptixDeviceContext context )
```

Destroys all CPU and GPU state associated with the device.

It will attempt to block on CUDA streams that have launch work outstanding.

Any API objects, such as OptixModule and OptixPipeline, not already destroyed will be destroyed.

Thread safety: A device context must not be destroyed while it is still in use by concurrent API calls in other threads.

8.11.2.19 optixDeviceContextGetCacheDatabaseSizes()

```
OptixResult optixDeviceContextGetCacheDatabaseSizes (
          OptixDeviceContext context,
          size_t * lowWaterMark,
          size_t * highWaterMark )
```

Returns the low and high water marks for disk cache garbage collection. If the cache has been disabled by setting the environment variable OPTIX_CACHE_MAXSIZE=0, this function will return 0 for the low and high water marks.

Parameters

in	*context*	the device context
out	*lowWaterMark*	the low water mark
out	*highWaterMark*	the high water mark

8.11.2.20 optixDeviceContextGetCacheEnabled()

```
OptixResult optixDeviceContextGetCacheEnabled (
          OptixDeviceContext context,
          int * enabled )
```

Indicates whether the disk cache is enabled or disabled.

Parameters

in	*context*	the device context
out	*enabled*	1 if enabled, 0 if disabled

8.11.2.21 optixDeviceContextGetCacheLocation()

```
OptixResult optixDeviceContextGetCacheLocation (
          OptixDeviceContext context,
          char * location,
          size_t locationSize )
```

Returns the location of the disk cache. If the cache has been disabled by setting the environment variable OPTIX_CACHE_MAXSIZE=0, this function will return an empy string.

Parameters

in	*context*	the device context
out	*location*	directory of disk cache, null terminated if locationSize > 0
in	*locationSize*	locationSize

8.11.2.22 optixDeviceContextGetProperty()

```
OptixResult optixDeviceContextGetProperty (
            OptixDeviceContext context,
            OptixDeviceProperty property,
            void * value,
            size_t sizeInBytes )
```

Query properties of a device context.

Parameters

in	*context*	the device context to query the property for
in	*property*	the property to query
out	*value*	pointer to the returned
in	*sizeInBytes*	size of output

8.11.2.23 optixDeviceContextSetCacheDatabaseSizes()

```
OptixResult optixDeviceContextSetCacheDatabaseSizes (
            OptixDeviceContext context,
            size_t lowWaterMark,
            size_t highWaterMark )
```

Sets the low and high water marks for disk cache garbage collection.

Garbage collection is triggered when a new entry is written to the cache and the current cache data size plus the size of the cache entry that is about to be inserted exceeds the high water mark. Garbage collection proceeds until the size reaches the low water mark. Garbage collection will always free enough space to insert the new entry without exceeding the low water mark. Setting either limit to zero will disable garbage collection. An error will be returned if both limits are non-zero and the high water mark is smaller than the low water mark.

Note that garbage collection is performed only on writes to the disk cache. No garbage collection is triggered on disk cache initialization or immediately when calling this function, but on subsequent inserting of data into the database.

If the size of a compiled module exceeds the value configured for the high water mark and garbage collection is enabled, the module will not be added to the cache and a warning will be added to the log.

The high water mark can be overridden with the environment variable OPTIX_CACHE_MAXSIZE. The environment variable takes precedence over the function parameters. The low water mark will be set to half the value of OPTIX_CACHE_MAXSIZE. Setting OPTIX_CACHE_MAXSIZE to 0 will disable the disk cache, but will not alter the contents of the cache. Negative and non-integer values will be ignored.

Parameters

in	*context*	the device context
in	*lowWaterMark*	the low water mark
in	*highWaterMark*	the high water mark

8.11.2.24 optixDeviceContextSetCacheEnabled()

```
OptixResult optixDeviceContextSetCacheEnabled (
            OptixDeviceContext context,
            int enabled )
```

Enables or disables the disk cache.

If caching was previously disabled, enabling it will attempt to initialize the disk cache database using the currently configured cache location. An error will be returned if initialization fails.

Note that no in-memory cache is used, so no caching behavior will be observed if the disk cache is disabled.

The cache can be disabled by setting the environment variable OPTIX_CACHE_MAXSIZE=0. The environment variable takes precedence over this setting. See optixDeviceContextSetCacheDatabaseSizes for additional information.

Note that the disk cache can be disabled by the environment variable, but it cannot be enabled via the environment if it is disabled via the API.

Parameters

in	*context*	the device context
in	*enabled*	1 to enabled, 0 to disable

8.11.2.25 optixDeviceContextSetCacheLocation()

```
OptixResult optixDeviceContextSetCacheLocation (
            OptixDeviceContext context,
            const char * location )
```

Sets the location of the disk cache.

The location is specified by a directory. This directory should not be used for other purposes and will be created if it does not exist. An error will be returned if 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	*context*	the device context
in	*location*	directory of disk cache

8.11.2.26 optixDeviceContextSetLogCallback()

```
OptixResult optixDeviceContextSetLogCallback (
            OptixDeviceContext context,
            OptixLogCallback callbackFunction,
            void * callbackData,
            unsigned int callbackLevel )
```

Sets the current log callback method.

See OptixLogCallback for more details.

Thread safety: It is guaranteed that the callback itself (callbackFunction and callbackData) are updated atomically. It is not guaranteed that the callback itself (callbackFunction and callbackData) and the callbackLevel are updated atomically. It is unspecified when concurrent API calls using the same context start to make use of the new callback method.

Parameters

in	*context*	the device context
in	*callbackFunction*	the callback function to call
in	*callbackData*	pointer to data passed to callback function while invoking it
in	*callbackLevel*	callback level

8.11.2.27 optixGetErrorName()

```
const char * optixGetErrorName (
            OptixResult result )
```

Returns a string containing the name of an error code in the enum.

Output is a string representation of the enum. For example "OPTIX_SUCCESS" for OPTIX_SUCCESS and "OPTIX_ERROR_INVALID_VALUE" for OPTIX_ERROR_INVALID_VALUE.

If the error code is not recognized, "Unrecognized OptixResult code" is returned.

Parameters

in	*result*	OptixResult enum to generate string name for

See also optixGetErrorString

8.11.2.28 optixGetErrorString()

```
const char * optixGetErrorString (
            OptixResult result )
```

Returns the description string for an error code.

Output is a string description of the enum. For example "Success" for OPTIX_SUCCESS and "Invalid value" for OPTIX_ERROR_INVALID_VALUE.

If the error code is not recognized, "Unrecognized OptixResult code" is returned.

Parameters

in	*result*	OptixResult enum to generate string description for

See also optixGetErrorName

8.11.2.29 optixLaunch()

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

Where the magic happens.

The stream and pipeline must belong to the same device context. Multiple launches may be issues in parallel from multiple threads to different streams.

pipelineParamsSize number of bytes are copied from the device memory pointed to by pipelineParams before launch. It is an error if pipelineParamsSize is greater than the size of the variable declared in modules and identified by OptixPipelineCompileOptions::pipelineLaunchParamsVariableName. If the launch params variable was optimized out or not found in the modules linked to the pipeline then the pipelineParams and pipelineParamsSize parameters are ignored.

sbt points to the shader binding table, which defines shader groupings and their resources. See the SBT spec.

Parameters

in	*pipeline*	
in	*stream*	
in	*pipelineParams*	
in	*pipelineParamsSize*	
in	*sbt*	
in	*width*	number of elements to compute
in	*height*	number of elements to compute
in	*depth*	number of elements to compute

Thread safety: In the current implementation concurrent launches to the same pipeline are not supported. Concurrent launches require separate OptixPipeline objects.

8.11.2.30 optixModuleCreateFromPTX()

```
OptixResult optixModuleCreateFromPTX (
            OptixDeviceContext context,
            const OptixModuleCompileOptions * moduleCompileOptions,
            const OptixPipelineCompileOptions * pipelineCompileOptions,
            const char * PTX,
            size_t PTXsize,
            char * logString,
            size_t * logStringSize,
            OptixModule * module )
```

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	*moduleCompileOptions*	
in	*pipelineCompileOptions*	All modules in a pipeline need to use the same values for the pipeline compile options.
in	*PTX*	Pointer to the PTX input string.
in	*PTXsize*	Parsing proceeds up to PTXsize characters, or the first NUL byte, whichever occurs first.
out	*logString*	Information will be written to this string. If logStringSize > 0 logString will be null terminated.
in,out	*logStringSize*	
out	*module*	

Returns

OPTIX_ERROR_INVALID_VALUE - context is 0, moduleCompileOptions is 0, pipelineCompileOptions is 0, PTX is 0, module is 0.

8.11.2.31 optixModuleCreateFromPTXWithTasks()

```
OptixResult optixModuleCreateFromPTXWithTasks (
            OptixDeviceContext context,
            const OptixModuleCompileOptions * moduleCompileOptions,
            const OptixPipelineCompileOptions * pipelineCompileOptions,
```

```
            const char * PTX,
            size_t PTXsize,
            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 optixModuleCreateFromPTX( ), aside from the return parameter, firstTask.

The memory used to hold the PTX 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 optixModuleCreateFromPTXWithTasks( ), it will contain all the compiler feedback from all executed tasks. The lifetime of the memory pointed to by logString should extend from calling optixModuleCreateFromPTXWithTasks( ) 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 optixModuleCreateFromPTXWithTasks( ) 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_IMPENDIND_FAILURE, any unstarted OptixTask objects do not need to be executed though there is no harm doing so.

See also optixModuleCreateFromPTX

8.11.2.32 optixModuleDestroy( )

```
OptixResult optixModuleDestroy (
            OptixModule module )
```

Call for OptixModule objects created with optixModuleCreateFromPTX and optixModuleDeserialize.

Modules must not be destroyed while they are still used by any program group.

Thread safety: A module must not be destroyed while it is still in use by concurrent API calls in other threads.

8.11.2.33 optixModuleGetCompilationState( )

```
OptixResult optixModuleGetCompilationState (
            OptixModule module,
            OptixModuleCompileState * state )
```

When creating a module with tasks, the current state of the module can be queried using this function.

Thread safety: Safe to call from any thread until optixModuleDestroy is called.

See also optixModuleCreateFromPTXWithTasks

8.11.2.34 optixOpacityMicromapArrayBuild()

```
OptixResult optixOpacityMicromapArrayBuild (
            OptixDeviceContext context,
            CUstream stream,
            const OptixOpacityMicromapArrayBuildInput * buildInput,
            const OptixMicromapBuffers * buffers )
```

Construct an array of Opacity Micromaps.

Each triangle within an instance/GAS may reference one opacity micromap to give finer control over alpha behavior. A opacity micromap consists of a set of 4^N micro-triangles in a triangular uniform barycentric grid. Multiple opacity micromaps are collected (built) into a opacity micromap array with this function. Each geometry in a GAS may bind a single opacity micromap array and can use opacity micromaps from that array only.

Each micro-triangle within a opacity micromap can be in one of four states: Transparent, Opaque, Unknown-Transparent or Unknown-Opaque. During traversal, if a triangle with a opacity micromap attached is intersected, the opacity micromap is queried to categorize the hit as either opaque, unknown (alpha) or a miss. Geometry, ray or instance flags that modify the alpha/opaque behavior are applied *after* this opacity micromap query.

The opacity micromap query may operate in 2-state mode (alpha testing) or 4-state mode (AHS culling), depending on the opacity micromap type and ray/instance flags. When operating in 2-state mode, alpha hits will not be reported, and transparent and opaque hits must be accurate.

Parameters

in	*context*	
in	*stream*	
in	*buildInput*	a single build input object referencing many opacity micromaps
in	*buffers*	the buffers used for build
	[in/out]	emittedProperties types of requested properties and output buffers
in	*numEmittedProperties*	number of post-build properties to populate (may be zero)

8.11.2.35 optixOpacityMicromapArrayComputeMemoryUsage()

```
OptixResult optixOpacityMicromapArrayComputeMemoryUsage (
            OptixDeviceContext context,
            const OptixOpacityMicromapArrayBuildInput * buildInput,
            OptixMicromapBufferSizes * bufferSizes )
```

Determine the amount of memory necessary for a Opacity Micromap Array build.

Parameters

in	*context*

Parameters

in	*buildInput*
out	*bufferSizes*

8.11.2.36 optixOpacityMicromapArrayGetRelocationInfo()

```
OptixResult optixOpacityMicromapArrayGetRelocationInfo (
            OptixDeviceContext context,
            CUdeviceptr opacityMicromapArray,
            OptixRelocationInfo * info )
```

Obtain relocation information, stored in OptixRelocationInfo, for a given context and opacity micromap array.

The relocation information can be passed to optixCheckRelocationCompatibility to determine if a opacity micromap array, referenced by buffers, can be relocated to a different device's memory space (see optixCheckRelocationCompatibility).

When used with optixOpacityMicromapArrayRelocate, it provides data necessary for doing the relocation.

If the opacity micromap array data associated with 'opacityMicromapArray' is copied multiple times, the same OptixRelocationInfo can also be used on all copies.

Parameters

in	*context*
in	*opacityMicromapArray*
out	*info*

8.11.2.37 optixOpacityMicromapArrayRelocate()

```
OptixResult optixOpacityMicromapArrayRelocate (
            OptixDeviceContext context,
            CUstream stream,
            const OptixRelocationInfo * info,
            CUdeviceptr targetOpacityMicromapArray,
            size_t targetOpacityMicromapArraySizeInBytes )
```

optixOpacityMicromapArrayRelocate is called to update the opacity micromap array after it has been relocated. Relocation is necessary when the opacity micromap array's location in device memory has changed. optixOpacityMicromapArrayRelocate does not copy the memory. This function only operates on the relocated memory whose new location is specified by 'targetOpacityMicromapArray'. The original memory (source) is not required to be valid, only the OptixRelocationInfo.

Before calling optixOpacityMicromapArrayRelocate, optixCheckRelocationCompatibility should be called to ensure the copy will be compatible with the destination device context.

The memory pointed to by 'targetOpacityMicromapArray' should be allocated with the same size as the source opacity micromap array. Similar to the 'OptixMicromapBuffers::output' used in optixOpacityMicromapArrayBuild, this pointer must be a multiple of OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT.

The memory in 'targetOpacityMicromapArray' must be allocated as long as the opacity micromap array is in use.

Note that any Acceleration Structures build using the original memory (source) as input will still be associated with this original memory. To associate an existing (possibly relocated) Acceleration Structures with the relocated opacity micromap array, use optixAccelBuild to update the existing Acceleration Structures (See OPTIX_BUILD_OPERATION_UPDATE)

Parameters

in	*context*
in	*stream*
in	*info*
in	*targetOpacityMicromapArray*
in	*targetOpacityMicromapArraySizeInBytes*

8.11.2.38 optixPipelineCreate()

```
OptixResult optixPipelineCreate (
            OptixDeviceContext context,
            const OptixPipelineCompileOptions * pipelineCompileOptions,
            const OptixPipelineLinkOptions * pipelineLinkOptions,
            const OptixProgramGroup * programGroups,
            unsigned int numProgramGroups,
            char * logString,
            size_t * logStringSize,
            OptixPipeline * pipeline )
```

logString is an optional buffer that contains compiler feedback and errors. This information is also passed to the context logger (if enabled), however it may be difficult to correlate output to the logger to specific API invocations when using multiple threads. The output to logString will only contain feedback for this specific invocation of this API call.

logStringSize as input should be a pointer to the number of bytes backing logString. Upon return it contains the length of the log message (including the null terminator) which may be greater than the input value. In this case, the log message will be truncated to fit into logString.

If logString or logStringSize are NULL, no output is written to logString. If logStringSize points to a value that is zero, no output is written. This does not affect output to the context logger if enabled.

Parameters

in	*context*	
in	*pipelineCompileOptions*	
in	*pipelineLinkOptions*	
in	*programGroups*	array of ProgramGroup objects
in	*numProgramGroups*	number of ProgramGroup objects
out	*logString*	Information will be written to this string. If logStringSize > 0 logString will be null terminated.
in,out	*logStringSize*	

Parameters

out	*pipeline*	

8.11.2.39 optixPipelineDestroy()

```
OptixResult optixPipelineDestroy (
          OptixPipeline pipeline )
```

Thread safety: A pipeline must not be destroyed while it is still in use by concurrent API calls in other threads.

8.11.2.40 optixPipelineSetStackSize()

```
OptixResult optixPipelineSetStackSize (
          OptixPipeline pipeline,
          unsigned int directCallableStackSizeFromTraversal,
          unsigned int directCallableStackSizeFromState,
          unsigned int continuationStackSize,
          unsigned int maxTraversableGraphDepth )
```

Sets the stack sizes for a pipeline.

Users are encouraged to see the programming guide and the implementations of the helper functions to understand how to construct the stack sizes based on their particular needs.

If this method is not used, an internal default implementation is used. The default implementation is correct (but not necessarily optimal) as long as the maximum depth of call trees of CC and DC programs is at most 2 and no motion transforms are used.

The maxTraversableGraphDepth responds to the maximal number of traversables visited when calling trace. Every acceleration structure and motion transform count as one level of traversal. E.g., for a simple IAS (instance acceleration structure) -> GAS (geometry acceleration structure) traversal graph, the maxTraversableGraphDepth is two. For IAS -> MT (motion transform) -> GAS, the maxTraversableGraphDepth is three. Note that it does not matter whether a IAS or GAS has motion or not, it always counts as one. Launching optix with exceptions turned on (see OPTIX_EXCEPTION_FLAG_TRACE_DEPTH) will throw an exception if the specified maxTraversableGraphDepth is too small.

Parameters

in	*pipeline*	The pipeline to configure the stack size for.
in	*directCallableStackSizeFromTraversal*	The direct stack size requirement for direct callables invoked from IS or AH.
in	*directCallableStackSizeFromState*	The direct stack size requirement for direct callables invoked from RG, MS, or CH.
in	*continuationStackSize*	The continuation stack requirement.
in	*maxTraversableGraphDepth*	The maximum depth of a traversable graph passed to trace.

8.11.2.41 optixProgramGroupCreate()

```
OptixResult optixProgramGroupCreate (
```

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

logString is an optional buffer that contains compiler feedback and errors. This information is also passed to the context logger (if enabled), however it may be difficult to correlate output to the logger to specific API invocations when using multiple threads. The output to logString will only contain feedback for this specific invocation of this API call.

logStringSize as input should be a pointer to the number of bytes backing logString. Upon return it contains the length of the log message (including the null terminator) which may be greater than the input value. In this case, the log message will be truncated to fit into logString.

If logString or logStringSize are NULL, no output is written to logString. If logStringSize points to a value that is zero, no output is written. This does not affect output to the context logger if enabled.

Creates numProgramGroups OptiXProgramGroup objects from the specified OptixProgramGroupDesc array. The size of the arrays must match.

Parameters

in	*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*	
out	*programGroups*	

8.11.2.42 optixProgramGroupDestroy()

```
OptixResult optixProgramGroupDestroy (
          OptixProgramGroup programGroup )
```

Thread safety: A program group must not be destroyed while it is still in use by concurrent API calls in other threads.

8.11.2.43 optixProgramGroupGetStackSize()

```
OptixResult optixProgramGroupGetStackSize (
          OptixProgramGroup programGroup,
          OptixStackSizes * stackSizes )
```

Returns the stack sizes for the given program group.

Parameters

in	*programGroup*	the program group
out	*stackSizes*	the corresponding stack sizes

8.11.2.44 optixSbtRecordPackHeader()

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

Parameters

in	*programGroup*	the program group containing the program(s)
out	*sbtRecordHeaderHostPointer*	the result sbt record header

8.11.2.45 optixTaskExecute()

```
OptixResult optixTaskExecute (
            OptixTask task,
            OptixTask * additionalTasks,
            unsigned int maxNumAdditionalTasks,
            unsigned int * numAdditionalTasksCreated )
```

Each OptixTask should be executed with optixTaskExecute( ). If additional parallel work is found, new OptixTask objects will be returned in additionalTasks along with the number of additional tasks in numAdditionalTasksCreated. The parameter additionalTasks should point to a user allocated array of minimum size maxNumAdditionalTasks. OptiX can generate upto maxNumAdditionalTasks additional tasks.

Each task can be executed in parallel and in any order.

Thread safety: Safe to call from any thread until optixModuleDestroy( ) is called for any associated task.

See also optixModuleCreateFromPTXWithTasks

Parameters

in	*task*	the OptixTask to execute
in	*additionalTasks*	pointer to array of OptixTask objects to be filled in
in	*maxNumAdditionalTasks*	maximum number of additional OptixTask objects
out	*numAdditionalTasksCreated*	number of OptixTask objects created by OptiX and written into #additionalTasks

8.12 optix_7_host.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
3 *
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
```

```
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly
8 * prohibited.
9 *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
27
28 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
29 #error("optix_7_host.h is an internal header file and must not be used directly.  Please use optix_host.h
or optix.h instead.")
30 #endif
31
32 #ifndef __optix_optix_7_host_h__
33 #define __optix_optix_7_host_h__
34
35 #include "optix_7_types.h"
36 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
37 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver types must be defined through other
38 // means before including optix headers.
39 #include <cuda.h>
40 #endif
41
42 #ifdef NV_MODULE_OPTIX
43 // driver build only, not visible in SDK
44 #include <exp/misc/optix_nvconfig_translate.h>  // includes <g_nvconfig.h>
45 #endif  // NV_MODULE_OPTIX
46
47
48 #ifdef __cplusplus
49 extern "C" {
50 #endif
51
54
57
58
69 const char* optixGetErrorName(OptixResult result);
70
81 const char* optixGetErrorString(OptixResult result);
82
84
86
87
106 OptixResult optixDeviceContextCreate(CUcontext fromContext, const OptixDeviceContextOptions* options,
OptixDeviceContext* context);
107
116 OptixResult optixDeviceContextDestroy(OptixDeviceContext context);
117
124 OptixResult optixDeviceContextGetProperty(OptixDeviceContext context, OptixDeviceProperty property,
void* value, size_t sizeInBytes);
125
139 OptixResult optixDeviceContextSetLogCallback(OptixDeviceContext context,
140                                               OptixLogCallback   callbackFunction,
141                                               void*              callbackData,
142                                               unsigned int       callbackLevel);
143
162 OptixResult optixDeviceContextSetCacheEnabled(OptixDeviceContext context,
163                                                int                enabled);
164
```

```
185 OptixResult optixDeviceContextSetCacheLocation(OptixDeviceContext context, const char* location);
186
214 OptixResult optixDeviceContextSetCacheDatabaseSizes(OptixDeviceContext context, size_t lowWaterMark,
size_t highWaterMark);
215
220 OptixResult optixDeviceContextGetCacheEnabled(OptixDeviceContext context, int* enabled);
227 OptixResult optixDeviceContextGetCacheLocation(OptixDeviceContext context, char* location, size_t
locationSize);
228
236 OptixResult optixDeviceContextGetCacheDatabaseSizes(OptixDeviceContext context, size_t* lowWaterMark,
size_t* highWaterMark);
237
239
241
242
266 OptixResult optixPipelineCreate(OptixDeviceContext                 context,
267                                 const OptixPipelineCompileOptions* pipelineCompileOptions,
268                                 const OptixPipelineLinkOptions*    pipelineLinkOptions,
269                                 const OptixProgramGroup*           programGroups,
270                                 unsigned int                       numProgramGroups,
271                                 char*                              logString,
272                                 size_t*                            logStringSize,
273                                 OptixPipeline*                     pipeline);
274
276 OptixResult optixPipelineDestroy(OptixPipeline pipeline);
277
300 OptixResult optixPipelineSetStackSize(OptixPipeline pipeline,
301                                       unsigned int  directCallableStackSizeFromTraversal,
302                                       unsigned int  directCallableStackSizeFromState,
303                                       unsigned int  continuationStackSize,
304                                       unsigned int  maxTraversableGraphDepth);
305
307
309
310
336 OptixResult optixModuleCreateFromPTX(OptixDeviceContext                 context,
337                                      const OptixModuleCompileOptions*   moduleCompileOptions,
338                                      const OptixPipelineCompileOptions* pipelineCompileOptions,
339                                      const char*                        PTX,
340                                      size_t                             PTXsize,
341                                      char*                              logString,
342                                      size_t*                            logStringSize,
343                                      OptixModule*                       module);
344
364
377
385 OptixResult optixModuleCreateFromPTXWithTasks(OptixDeviceContext                 context,
386                                               const OptixModuleCompileOptions*   moduleCompileOptions,
387                                               const OptixPipelineCompileOptions* pipelineCompileOptions,
388                                               const char*                        PTX,
389                                               size_t                             PTXsize,
390                                               char*                              logString,
391                                               size_t*                            logStringSize,
392                                               OptixModule*                       module,
393                                               OptixTask*                         firstTask);
394
401 OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState* state);
402
408 OptixResult optixModuleDestroy(OptixModule module);
409
413 OptixResult optixBuiltinISModuleGet(OptixDeviceContext                 context,
414                                     const OptixModuleCompileOptions*   moduleCompileOptions,
415                                     const OptixPipelineCompileOptions* pipelineCompileOptions,
416                                     const OptixBuiltinISOptions*       builtinISOptions,
417                                     OptixModule*                       builtinModule);
418
420
```

```
422
423
441 OptixResult optixTaskExecute(OptixTask task, OptixTask* additionalTasks, unsigned int
maxNumAdditionalTasks, unsigned int* numAdditionalTasksCreated);
442
444
446
447
452 OptixResult optixProgramGroupGetStackSize(OptixProgramGroup programGroup, OptixStackSizes* stackSizes);
453
479 OptixResult optixProgramGroupCreate(OptixDeviceContext             context,
480                                     const OptixProgramGroupDesc*    programDescriptions,
481                                     unsigned int                    numProgramGroups,
482                                     const OptixProgramGroupOptions* options,
483                                     char*                           logString,
484                                     size_t*                         logStringSize,
485                                     OptixProgramGroup*              programGroups);
486
488 OptixResult optixProgramGroupDestroy(OptixProgramGroup programGroup);
489
491
493
494
521 OptixResult optixLaunch(OptixPipeline                 pipeline,
522                         CUstream                      stream,
523                         CUdeviceptr                   pipelineParams,
524                         size_t                        pipelineParamsSize,
525                         const OptixShaderBindingTable* sbt,
526                         unsigned int                  width,
527                         unsigned int                  height,
528                         unsigned int                  depth);
529
532 OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void* sbtRecordHeaderHostPointer);
533
535
537
538
544 OptixResult optixAccelComputeMemoryUsage(OptixDeviceContext           context,
545                                          const OptixAccelBuildOptions* accelOptions,
546                                          const OptixBuildInput*        buildInputs,
547                                          unsigned int                  numBuildInputs,
548                                          OptixAccelBufferSizes*        bufferSizes);
549
562 OptixResult optixAccelBuild(OptixDeviceContext            context,
563                             CUstream                      stream,
564                             const OptixAccelBuildOptions* accelOptions,
565                             const OptixBuildInput*        buildInputs,
566                             unsigned int                  numBuildInputs,
567                             CUdeviceptr                   tempBuffer,
568                             size_t                        tempBufferSizeInBytes,
569                             CUdeviceptr                   outputBuffer,
570                             size_t                        outputBufferSizeInBytes,
571                             OptixTraversableHandle*       outputHandle,
572                             const OptixAccelEmitDesc*     emittedProperties,
573                             unsigned int                  numEmittedProperties);
574
592 OptixResult optixAccelGetRelocationInfo(OptixDeviceContext context, OptixTraversableHandle handle,
OptixRelocationInfo* info);
593
605 OptixResult optixCheckRelocationCompatibility(OptixDeviceContext context, const OptixRelocationInfo*
info, int* compatible);
606
644 OptixResult optixAccelRelocate(OptixDeviceContext         context,
645                                CUstream                   stream,
646                                const OptixRelocationInfo* info,
647                                const OptixRelocateInput*  relocateInputs,
648                                size_t                     numRelocateInputs,
```

```
649                                 CUdeviceptr               targetAccel,
650                                 size_t                    targetAccelSizeInBytes,
651                                 OptixTraversableHandle*   targetHandle);
652
670 OptixResult optixAccelCompact(OptixDeviceContext      context,
671                                 CUstream                stream,
672                                 OptixTraversableHandle  inputHandle,
673                                 CUdeviceptr             outputBuffer,
674                                 size_t                  outputBufferSizeInBytes,
675                                 OptixTraversableHandle* outputHandle);
676
681 OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext      onDevice,
682                                                    CUdeviceptr             pointer,
683                                                    OptixTraversableType    traversableType,
684                                                    OptixTraversableHandle* traversableHandle);
685
686
692 OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext                      context,
693                                                         const OptixOpacityMicromapArrayBuildInput* buildInput,
694                                                         OptixMicromapBufferSizes*                  bufferSizes);
695
720 OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext                       context,
721                                            CUstream                                 stream,
722                                            const OptixOpacityMicromapArrayBuildInput* buildInput,
723                                            const OptixMicromapBuffers*              buffers);
724
740 OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext context, CUdeviceptr
opacityMicromapArray, OptixRelocationInfo* info);
741
768 OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext         context,
769                                               CUstream                   stream,
770                                               const OptixRelocationInfo* info,
771                                               CUdeviceptr                targetOpacityMicromapArray,
772                                               size_t
targetOpacityMicromapArraySizeInBytes);
773
774
775
777
779
780
792 OptixResult optixDenoiserCreate(OptixDeviceContext context,
793                                 OptixDenoiserModelKind modelKind,
794                                 const OptixDenoiserOptions* options,
795                                 OptixDenoiser* denoiser);
796
809 OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context,
810                                              const void* userData, size_t userDataSizeInBytes,
OptixDenoiser* denoiser);
811
813 OptixResult optixDenoiserDestroy(OptixDenoiser denoiser);
814
834 OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser denoiser,
835                                                 unsigned int        outputWidth,
836                                                 unsigned int        outputHeight,
837                                                 OptixDenoiserSizes* returnSizes);
838
855 OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
856                                CUstream      stream,
857                                unsigned int  inputWidth,
858                                unsigned int  inputHeight,
859                                CUdeviceptr   denoiserState,
860                                size_t        denoiserStateSizeInBytes,
861                                CUdeviceptr   scratch,
862                                size_t        scratchSizeInBytes);
863
925 OptixResult optixDenoiserInvoke(OptixDenoiser                 denoiser,
```

```
926                                     CUstream                        stream,
927                                     const OptixDenoiserParams*      params,
928                                     CUdeviceptr                     denoiserState,
929                                     size_t                          denoiserStateSizeInBytes,
930                                     const OptixDenoiserGuideLayer*  guideLayer,
931                                     const OptixDenoiserLayer*       layers,
932                                     unsigned int                    numLayers,
933                                     unsigned int                    inputOffsetX,
934                                     unsigned int                    inputOffsetY,
935                                     CUdeviceptr                     scratch,
936                                     size_t                          scratchSizeInBytes);
937
961 OptixResult optixDenoiserComputeIntensity(OptixDenoiser       denoiser,
962                                           CUstream            stream,
963                                           const OptixImage2D* inputImage,
964                                           CUdeviceptr         outputIntensity,
965                                           CUdeviceptr         scratch,
966                                           size_t              scratchSizeInBytes);
967
982 OptixResult optixDenoiserComputeAverageColor(OptixDenoiser       denoiser,
983                                              CUstream            stream,
984                                              const OptixImage2D* inputImage,
985                                              CUdeviceptr         outputAverageColor,
986                                              CUdeviceptr         scratch,
987                                              size_t              scratchSizeInBytes);
988
990
991 #ifdef __cplusplus
992 }
993 #endif
994
995 #include "optix_function_table.h"
996
997 #endif  // __optix_optix_7_host_h__
```

8.13 optix_7_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 OptixImage2D
- struct OptixDenoiserOptions
- struct OptixDenoiserGuideLayer
- struct OptixDenoiserLayer
- struct OptixDenoiserParams
- struct OptixDenoiserSizes
- struct OptixModuleCompileBoundValueEntry
- struct OptixPayloadType
- struct OptixModuleCompileOptions
- struct OptixProgramGroupSingleModule
- struct OptixProgramGroupHitgroup
- struct OptixProgramGroupCallables
- struct OptixProgramGroupDesc
- struct OptixProgramGroupOptions
- struct OptixPipelineCompileOptions
- struct OptixPipelineLinkOptions
- struct OptixShaderBindingTable
- struct OptixStackSizes
- struct OptixBuiltinISOptions

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_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
- #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT (0)
- #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE (1)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT (2)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE (3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT (-1)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE (-2)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT (-3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE (-4)
- #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
- #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12

Typedefs

- typedef unsigned long long CUdeviceptr
- typedef struct OptixDeviceContext_t * OptixDeviceContext
- typedef struct OptixModule_t * OptixModule
- typedef struct OptixProgramGroup_t * OptixProgramGroup
- typedef struct OptixPipeline_t * OptixPipeline
- typedef struct OptixDenoiser_t * OptixDenoiser
- typedef struct OptixTask_t * OptixTask
- typedef unsigned long long OptixTraversableHandle
- typedef unsigned int OptixVisibilityMask
- typedef enum OptixResult OptixResult
- typedef enum OptixDeviceProperty OptixDeviceProperty
- typedef void(* OptixLogCallback) (unsigned int level, const char *tag, const char *message, void *cbdata)
- typedef enum OptixDeviceContextValidationMode OptixDeviceContextValidationMode
- typedef struct OptixDeviceContextOptions OptixDeviceContextOptions
- typedef enum OptixGeometryFlags OptixGeometryFlags
- typedef enum OptixHitKind OptixHitKind
- typedef enum OptixIndicesFormat OptixIndicesFormat
- typedef enum OptixVertexFormat OptixVertexFormat
- typedef enum OptixTransformFormat OptixTransformFormat
- typedef enum 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 OptixPixelFormat OptixPixelFormat
- typedef struct OptixImage2D OptixImage2D
- typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
- typedef struct OptixDenoiserOptions OptixDenoiserOptions
- typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
- typedef struct OptixDenoiserLayer OptixDenoiserLayer
- typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
- typedef struct OptixDenoiserParams OptixDenoiserParams
- typedef struct OptixDenoiserSizes OptixDenoiserSizes
- typedef enum OptixRayFlags OptixRayFlags
- typedef enum OptixTransformType OptixTransformType
- typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags
- typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
- typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
- typedef enum OptixModuleCompileState OptixModuleCompileState
- typedef struct OptixModuleCompileBoundValueEntry OptixModuleCompileBoundValueEntry
- typedef enum OptixPayloadTypeID OptixPayloadTypeID
- typedef enum OptixPayloadSemantics OptixPayloadSemantics
- typedef struct OptixPayloadType OptixPayloadType
- typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
- typedef enum OptixProgramGroupKind OptixProgramGroupKind
- typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
- typedef struct OptixProgramGroupSingleModule OptixProgramGroupSingleModule
- typedef struct OptixProgramGroupHitgroup OptixProgramGroupHitgroup
- typedef struct OptixProgramGroupCallables OptixProgramGroupCallables
- typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
- typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
- typedef enum OptixExceptionCodes OptixExceptionCodes
- typedef enum OptixExceptionFlags OptixExceptionFlags
- typedef struct OptixPipelineCompileOptions OptixPipelineCompileOptions
- typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
- typedef struct OptixShaderBindingTable OptixShaderBindingTable
- typedef struct OptixStackSizes OptixStackSizes
- typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
- typedef OptixResult() OptixQueryFunctionTable_t(int abiId, unsigned int numOptions, OptixQueryFunctionTableOptions *, const void **, void *functionTable, size_t sizeOfTable)
- typedef struct OptixBuiltinISOptions OptixBuiltinISOptions

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_PTX = 7200 ,
  OPTIX_ERROR_INVALID_LAUNCH_PARAMETER = 7201 ,
  OPTIX_ERROR_INVALID_PAYLOAD_ACCESS = 7202 ,
  OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS = 7203 ,
  OPTIX_ERROR_INVALID_FUNCTION_USE = 7204 ,
  OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS = 7205 ,
  OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250 ,
  OPTIX_ERROR_PIPELINE_LINK_ERROR = 7251 ,
  OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE = 7270 ,
  OPTIX_ERROR_INTERNAL_COMPILER_ERROR = 7299 ,
  OPTIX_ERROR_DENOISER_MODEL_NOT_SET = 7300 ,
  OPTIX_ERROR_DENOISER_NOT_INITIALIZED = 7301 ,
  OPTIX_ERROR_NOT_COMPATIBLE = 7400 ,
  OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH = 7500 ,
  OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED = 7501 ,
  OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID = 7502 ,
  OPTIX_ERROR_NOT_SUPPORTED = 7800 ,
  OPTIX_ERROR_UNSUPPORTED_ABI_VERSION = 7801 ,
  OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH = 7802 ,
  OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS = 7803 ,
  OPTIX_ERROR_LIBRARY_NOT_FOUND = 7804 ,
  OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND = 7805 ,
  OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE = 7806 ,
  OPTIX_ERROR_DEVICE_OUT_OF_MEMORY = 7807 ,
  OPTIX_ERROR_CUDA_ERROR = 7900 ,
  OPTIX_ERROR_INTERNAL_ERROR = 7990 ,
  OPTIX_ERROR_UNKNOWN = 7999 }
- enum OptixDeviceProperty {
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004 ,
  OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009 }

- enum OptixDeviceContextValidationMode {
  OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0 ,
  OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF }
- enum OptixGeometryFlags {
  OPTIX_GEOMETRY_FLAG_NONE = 0 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
  OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u << 1 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 2 }
- enum OptixHitKind {
  OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE ,
  OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF }
- enum OptixIndicesFormat {
  OPTIX_INDICES_FORMAT_NONE = 0 ,
  OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102 ,
  OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103 }
- enum OptixVertexFormat {
  OPTIX_VERTEX_FORMAT_NONE = 0 ,
  OPTIX_VERTEX_FORMAT_FLOAT3 = 0x2121 ,
  OPTIX_VERTEX_FORMAT_FLOAT2 = 0x2122 ,
  OPTIX_VERTEX_FORMAT_HALF3 = 0x2123 ,
  OPTIX_VERTEX_FORMAT_HALF2 = 0x2124 ,
  OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125 ,
  OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126 }
- enum OptixTransformFormat {
  OPTIX_TRANSFORM_FORMAT_NONE = 0 ,
  OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1 }
- enum 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_SPHERE = 0x2506 ,
  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_SPHERE = 1 << 6 ,
  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 OptixPixelFormat {
  OPTIX_PIXEL_FORMAT_HALF2 = 0x2207 ,
  OPTIX_PIXEL_FORMAT_HALF3 = 0x2201 ,
  OPTIX_PIXEL_FORMAT_HALF4 = 0x2202 ,
  OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208 ,
  OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203 ,
  OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204 ,
  OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205 ,
  OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206 ,
  OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209 }

- enum OptixDenoiserModelKind {
  OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322 ,
  OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323 ,
  OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326 ,
  OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328 }
- enum OptixDenoiserAlphaMode {
  OPTIX_DENOISER_ALPHA_MODE_COPY = 0 ,
  OPTIX_DENOISER_ALPHA_MODE_ALPHA_AS_AOV = 1 ,
  OPTIX_DENOISER_ALPHA_MODE_FULL_DENOISE_PASS = 2 }
- enum OptixRayFlags {
  OPTIX_RAY_FLAG_NONE = 0u ,
  OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
  OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u << 1 ,
  OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u << 2 ,
  OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u << 3 ,
  OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u << 4 ,
  OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u << 5 ,
  OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u << 6 ,
  OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u << 7 ,
  OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 10 }
- enum OptixTransformType {
  OPTIX_TRANSFORM_TYPE_NONE = 0 ,
  OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM = 1 ,
  OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM = 2 ,
  OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM = 3 ,
  OPTIX_TRANSFORM_TYPE_INSTANCE = 4 }
- enum OptixTraversableGraphFlags {
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u << 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u << 1 }
- enum OptixCompileOptimizationLevel {
  OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343 }
- enum OptixCompileDebugLevel {
  OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT = 0 ,
  OPTIX_COMPILE_DEBUG_LEVEL_NONE = 0x2350 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL = 0x2351 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353 ,
  OPTIX_COMPILE_DEBUG_LEVEL_FULL = 0x2352 }
- enum OptixModuleCompileState {
  OPTIX_MODULE_COMPILE_STATE_NOT_STARTED = 0x2360 ,
  OPTIX_MODULE_COMPILE_STATE_STARTED = 0x2361 ,
  OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE = 0x2362 ,
  OPTIX_MODULE_COMPILE_STATE_FAILED = 0x2363 ,
  OPTIX_MODULE_COMPILE_STATE_COMPLETED = 0x2364 }
- enum OptixPayloadTypeID {
  OPTIX_PAYLOAD_TYPE_DEFAULT = 0 ,

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

- enum OptixPayloadSemantics {
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ = 1u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE = 2u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_READ = 1u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_WRITE = 2u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE = 3u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_READ = 1u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_WRITE = 2u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE = 3u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_READ = 1u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_WRITE = 2u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE = 3u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_READ = 1u << 8 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_WRITE = 2u << 8 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE = 3u << 8 }
- enum OptixProgramGroupKind {
  OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421 ,
  OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422 ,
  OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423 ,
  OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424 ,
  OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425 }
- enum OptixProgramGroupFlags { OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0 }
- enum OptixExceptionCodes {
  OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1 ,
  OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED = -3 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE = -5 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT = -6 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT = -7 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE = -8 ,
  OPTIX_EXCEPTION_CODE_INVALID_RAY = -9 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH = -10 ,
  OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH = -11 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT = -12 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD = -13 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD = -14 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS = -15 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0 = -16 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_1 = -17 ,

OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2 = -18 ,
OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS = -32 ,
OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH = -33 }

- enum OptixExceptionFlags {
  OPTIX_EXCEPTION_FLAG_NONE = 0 ,
  OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u << 0 ,
  OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u << 1 ,
  OPTIX_EXCEPTION_FLAG_USER = 1u << 2 ,
  OPTIX_EXCEPTION_FLAG_DEBUG = 1u << 3 }
- enum OptixQueryFunctionTableOptions { OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0 }

8.13.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

OptiX types include file – defines types and enums used by the API. For the math library routines include optix_math.h

8.14 optix_7_types.h

Go to the documentation of this file.

```
1
2 /*
3 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
4 *
5 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
6 * rights in and to this software, related documentation and any modifications thereto.
7 * Any use, reproduction, disclosure or distribution of this software and related
8 * documentation without an express license agreement from NVIDIA Corporation is strictly
9 * prohibited.
10 *
11 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
12 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
13 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
14 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
15 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
16 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
17 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
18 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
19 * SUCH DAMAGES
20 */
21
28
29 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
30 #error("optix_7_types.h is an internal header file and must not be used directly.  Please use
optix_types.h, optix_host.h, optix_device.h or optix.h instead.")
31 #endif
32
33 #ifndef __optix_optix_7_types_h__
34 #define __optix_optix_7_types_h__
35
36 #if !defined(__CUDACC_RTC__)
37 #include <stddef.h> /* for size_t */
38 #endif
39
40 #ifdef NV_MODULE_OPTIX
```

```
41 // driver build only, not visible in SDK
42 #include <exp/misc/optix_nvconfig_translate.h>  // includes <g_nvconfig.h>
43 #endif  // NV_MODULE_OPTIX
44
45
48
53 // This typedef should match the one in cuda.h in order to avoid compilation errors.
54 #if defined(_WIN64) || defined(__LP64__)
56 typedef unsigned long long CUdeviceptr;
57 #else
59 typedef unsigned int CUdeviceptr;
60 #endif
61
63 typedef struct OptixDeviceContext_t* OptixDeviceContext;
64
66 typedef struct OptixModule_t* OptixModule;
67
69 typedef struct OptixProgramGroup_t* OptixProgramGroup;
70
72 typedef struct OptixPipeline_t* OptixPipeline;
73
75 typedef struct OptixDenoiser_t* OptixDenoiser;
76
78 typedef struct OptixTask_t* OptixTask;
79
81 typedef unsigned long long OptixTraversableHandle;
82
84 typedef unsigned int OptixVisibilityMask;
85
87 #define OPTIX_SBT_RECORD_HEADER_SIZE ((size_t)32)
88
90 #define OPTIX_SBT_RECORD_ALIGNMENT 16ull
91
93 #define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128ull
94
96 #define OPTIX_INSTANCE_BYTE_ALIGNMENT 16ull
97
99 #define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8ull
100
102 #define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16ull
103
105 #define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
106
108 #define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
109
111 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
112
114 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
115
118 #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT          (0)
119 #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE               (1)
120 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT  (2)
121 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE       (3)
122
125 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT          (-1)
126 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE               (-2)
127 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT  (-3)
128 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE       (-4)
129
131 #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
132
134 #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
135
136
144 typedef enum OptixResult
145 {
146     OPTIX_SUCCESS                                = 0,
```

```
147     OPTIX_ERROR_INVALID_VALUE                   = 7001,
148     OPTIX_ERROR_HOST_OUT_OF_MEMORY              = 7002,
149     OPTIX_ERROR_INVALID_OPERATION               = 7003,
150     OPTIX_ERROR_FILE_IO_ERROR                   = 7004,
151     OPTIX_ERROR_INVALID_FILE_FORMAT             = 7005,
152     OPTIX_ERROR_DISK_CACHE_INVALID_PATH         = 7010,
153     OPTIX_ERROR_DISK_CACHE_PERMISSION_ERROR     = 7011,
154     OPTIX_ERROR_DISK_CACHE_DATABASE_ERROR       = 7012,
155     OPTIX_ERROR_DISK_CACHE_INVALID_DATA         = 7013,
156     OPTIX_ERROR_LAUNCH_FAILURE                  = 7050,
157     OPTIX_ERROR_INVALID_DEVICE_CONTEXT          = 7051,
158     OPTIX_ERROR_CUDA_NOT_INITIALIZED            = 7052,
159     OPTIX_ERROR_VALIDATION_FAILURE              = 7053,
160     OPTIX_ERROR_INVALID_PTX                     = 7200,
161     OPTIX_ERROR_INVALID_LAUNCH_PARAMETER        = 7201,
162     OPTIX_ERROR_INVALID_PAYLOAD_ACCESS          = 7202,
163     OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS        = 7203,
164     OPTIX_ERROR_INVALID_FUNCTION_USE            = 7204,
165     OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS      = 7205,
166     OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250,
167     OPTIX_ERROR_PIPELINE_LINK_ERROR             = 7251,
168     OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE     = 7270,
169     OPTIX_ERROR_INTERNAL_COMPILER_ERROR         = 7299,
170     OPTIX_ERROR_DENOISER_MODEL_NOT_SET          = 7300,
171     OPTIX_ERROR_DENOISER_NOT_INITIALIZED        = 7301,
172     OPTIX_ERROR_NOT_COMPATIBLE                  = 7400,
173     OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH           = 7500,
174     OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED  = 7501,
175     OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID         = 7502,
176     OPTIX_ERROR_NOT_SUPPORTED                   = 7800,
177     OPTIX_ERROR_UNSUPPORTED_ABI_VERSION         = 7801,
178     OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH    = 7802,
179     OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS  = 7803,
180     OPTIX_ERROR_LIBRARY_NOT_FOUND               = 7804,
181     OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND          = 7805,
182     OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE          = 7806,
183     OPTIX_ERROR_DEVICE_OUT_OF_MEMORY            = 7807,
184     OPTIX_ERROR_CUDA_ERROR                      = 7900,
185     OPTIX_ERROR_INTERNAL_ERROR                  = 7990,
186     OPTIX_ERROR_UNKNOWN                         = 7999,
187 } OptixResult;
188
192 typedef enum OptixDeviceProperty
193 {
195     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001,
196
199     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002,
200
203     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003,
204
207     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004,
208
211     OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005,
212
214     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006,
215
218     OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007,
219
222     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008,
223
225     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009,
226 } OptixDeviceProperty;
227
252 typedef void (*OptixLogCallback)(unsigned int level, const char* tag, const char* message, void* cbdata);
253
261 typedef enum OptixDeviceContextValidationMode
262 {
```

```
263     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0,
264     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF
265 } OptixDeviceContextValidationMode;
266
270 typedef struct OptixDeviceContextOptions
271 {
273     OptixLogCallback logCallbackFunction;
275     void* logCallbackData;
277     int logCallbackLevel;
279     OptixDeviceContextValidationMode validationMode;
280 } OptixDeviceContextOptions;
281
285 typedef enum OptixGeometryFlags
286 {
288     OPTIX_GEOMETRY_FLAG_NONE = 0,
289
292     OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u « 0,
293
297     OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u « 1,
298
302     OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u « 2,
303 } OptixGeometryFlags;
304
310 typedef enum OptixHitKind
311 {
313     OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE,
315     OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF
316 } OptixHitKind;
317
319 typedef enum OptixIndicesFormat
320 {
322     OPTIX_INDICES_FORMAT_NONE = 0,
324     OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102,
326     OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103
327 } OptixIndicesFormat;
328
330 typedef enum OptixVertexFormat
331 {
332     OPTIX_VERTEX_FORMAT_NONE      = 0,
333     OPTIX_VERTEX_FORMAT_FLOAT3    = 0x2121,
334     OPTIX_VERTEX_FORMAT_FLOAT2    = 0x2122,
335     OPTIX_VERTEX_FORMAT_HALF3     = 0x2123,
336     OPTIX_VERTEX_FORMAT_HALF2     = 0x2124,
337     OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125,
338     OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126
339 } OptixVertexFormat;
340
342 typedef enum OptixTransformFormat
343 {
344     OPTIX_TRANSFORM_FORMAT_NONE           = 0,
345     OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1,
346 } OptixTransformFormat;
347
348
350 typedef enum OptixOpacityMicromapFormat
351 {
353     OPTIX_OPACITY_MICROMAP_FORMAT_NONE = 0,
355     OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE = 1,
357     OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE = 2,
358 } OptixOpacityMicromapFormat;
359
361 typedef enum OptixOpacityMicromapArrayIndexingMode
362 {
364     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0,
367     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1,
371     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2,
372 } OptixOpacityMicromapArrayIndexingMode;
```

```
373
378 typedef struct OptixOpacityMicromapUsageCount
379 {
382     unsigned int count;
384     unsigned int subdivisionLevel;
386     OptixOpacityMicromapFormat format;
387 } OptixOpacityMicromapUsageCount;
388
389 typedef struct OptixBuildInputOpacityMicromap
390 {
392     OptixOpacityMicromapArrayIndexingMode indexingMode;
393
398     CUdeviceptr  opacityMicromapArray;
399
409     CUdeviceptr  indexBuffer;
410
413     unsigned int indexSizeInBytes;
414
417     unsigned int indexStrideInBytes;
418
420     unsigned int indexOffset;
421
423     unsigned int numMicromapUsageCounts;
426     const OptixOpacityMicromapUsageCount* micromapUsageCounts;
427 } OptixBuildInputOpacityMicromap;
428
429 typedef struct OptixRelocateInputOpacityMicromap
430 {
434     CUdeviceptr  opacityMicromapArray;
435 } OptixRelocateInputOpacityMicromap;
436
437
441 typedef struct OptixBuildInputTriangleArray
442 {
450     const CUdeviceptr* vertexBuffers;
451
453     unsigned int numVertices;
454
456     OptixVertexFormat vertexFormat;
457
460     unsigned int vertexStrideInBytes;
461
465     CUdeviceptr indexBuffer;
466
468     unsigned int numIndexTriplets;
469
471     OptixIndicesFormat indexFormat;
472
475     unsigned int indexStrideInBytes;
476
480     CUdeviceptr preTransform;
481
485     const unsigned int* flags;
486
488     unsigned int numSbtRecords;
489
493     CUdeviceptr sbtIndexOffsetBuffer;
494
496     unsigned int sbtIndexOffsetSizeInBytes;
497
500     unsigned int sbtIndexOffsetStrideInBytes;
501
504     unsigned int primitiveIndexOffset;
505
507     OptixTransformFormat transformFormat;
508
510     OptixBuildInputOpacityMicromap opacityMicromap;
```

```
511
512 } OptixBuildInputTriangleArray;
513
517 typedef struct OptixRelocateInputTriangleArray
518 {
521     unsigned int numSbtRecords;
522
524     OptixRelocateInputOpacityMicromap opacityMicromap;
525 } OptixRelocateInputTriangleArray;
526
529 typedef enum OptixPrimitiveType
530 {
532     OPTIX_PRIMITIVE_TYPE_CUSTOM                         = 0x2500,
534     OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE        = 0x2501,
536     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE            = 0x2502,
538     OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR                   = 0x2503,
540     OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM               = 0x2504,
541     OPTIX_PRIMITIVE_TYPE_SPHERE                         = 0x2506,
543     OPTIX_PRIMITIVE_TYPE_TRIANGLE                       = 0x2531,
544 } OptixPrimitiveType;
545
549 typedef enum OptixPrimitiveTypeFlags
550 {
552     OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM                   = 1 « 0,
554     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE  = 1 « 1,
556     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE      = 1 « 2,
558     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR             = 1 « 3,
560     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM         = 1 « 4,
561     OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE                   = 1 « 6,
563     OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE                 = 1 « 31,
564 } OptixPrimitiveTypeFlags;
565
568 typedef enum OptixCurveEndcapFlags
569 {
571     OPTIX_CURVE_ENDCAP_DEFAULT                        = 0,
573     OPTIX_CURVE_ENDCAP_ON                             = 1 « 0,
574 } OptixCurveEndcapFlags;
575
593 typedef struct OptixBuildInputCurveArray
594 {
597     OptixPrimitiveType curveType;
599     unsigned int numPrimitives;
600
605     const CUdeviceptr* vertexBuffers;
607     unsigned int numVertices;
610     unsigned int vertexStrideInBytes;
611
614     const CUdeviceptr* widthBuffers;
617     unsigned int widthStrideInBytes;
618
620     const CUdeviceptr* normalBuffers;
622     unsigned int normalStrideInBytes;
623
629     CUdeviceptr indexBuffer;
632     unsigned int indexStrideInBytes;
633
636     unsigned int flag;
637
640     unsigned int primitiveIndexOffset;
641
643     unsigned int endcapFlags;
644 } OptixBuildInputCurveArray;
645
658 typedef struct OptixBuildInputSphereArray
659 {
664   const CUdeviceptr* vertexBuffers;
665
```

```
668   unsigned int vertexStrideInBytes;
670   unsigned int numVertices;
671
674   const CUdeviceptr* radiusBuffers;
677   unsigned int radiusStrideInBytes;
680   int singleRadius;
681
685   const unsigned int* flags;
686
688   unsigned int numSbtRecords;
692   CUdeviceptr sbtIndexOffsetBuffer;
694   unsigned int sbtIndexOffsetSizeInBytes;
697   unsigned int sbtIndexOffsetStrideInBytes;
698
701   unsigned int primitiveIndexOffset;
702 } OptixBuildInputSphereArray;
703
705 typedef struct OptixAabb
706 {
707     float minX;
708     float minY;
709     float minZ;
710     float maxX;
711     float maxY;
712     float maxZ;
713 } OptixAabb;
714
718 typedef struct OptixBuildInputCustomPrimitiveArray
719 {
724     const CUdeviceptr* aabbBuffers;
725
728     unsigned int numPrimitives;
729
733     unsigned int strideInBytes;
734
738     const unsigned int* flags;
739
741     unsigned int numSbtRecords;
742
746     CUdeviceptr sbtIndexOffsetBuffer;
747
749     unsigned int sbtIndexOffsetSizeInBytes;
750
753     unsigned int sbtIndexOffsetStrideInBytes;
754
757     unsigned int primitiveIndexOffset;
758 } OptixBuildInputCustomPrimitiveArray;
759
763 typedef struct OptixBuildInputInstanceArray
764 {
772     CUdeviceptr instances;
773
775     unsigned int numInstances;
776
780     unsigned int instanceStride;
781 } OptixBuildInputInstanceArray;
782
786 typedef struct OptixRelocateInputInstanceArray
787 {
790     unsigned int numInstances;
791
797     CUdeviceptr traversableHandles;
798
799 } OptixRelocateInputInstanceArray;
800
804 typedef enum OptixBuildInputType
805 {
```

```
807     OPTIX_BUILD_INPUT_TYPE_TRIANGLES = 0x2141,
809     OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES = 0x2142,
811     OPTIX_BUILD_INPUT_TYPE_INSTANCES = 0x2143,
813     OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS = 0x2144,
815     OPTIX_BUILD_INPUT_TYPE_CURVES = 0x2145,
817     OPTIX_BUILD_INPUT_TYPE_SPHERES = 0x2146
818 } OptixBuildInputType;
819
825 typedef struct OptixBuildInput
826 {
828     OptixBuildInputType type;
829
830     union
831     {
833         OptixBuildInputTriangleArray triangleArray;
835         OptixBuildInputCurveArray curveArray;
837         OptixBuildInputSphereArray sphereArray;
839         OptixBuildInputCustomPrimitiveArray customPrimitiveArray;
841         OptixBuildInputInstanceArray instanceArray;
842         char pad[1024];
843     };
844 } OptixBuildInput;
845
849 typedef struct OptixRelocateInput
850 {
852     OptixBuildInputType type;
853
854     union
855     {
857         OptixRelocateInputInstanceArray instanceArray;
858
860         OptixRelocateInputTriangleArray triangleArray;
861
863     };
864 } OptixRelocateInput;
865
866 // Some 32-bit tools use this header. This static_assert fails for them because
867 // the default enum size is 4 bytes, rather than 8, under 32-bit compilers.
868 // This #ifndef allows them to disable the static assert.
869
870 // TODO Define a static assert for C/pre-C++-11
871 #if defined(__cplusplus) && __cplusplus >= 201103L
872 static_assert(sizeof(OptixBuildInput) == 8 + 1024, "OptixBuildInput has wrong size");
873 #endif
874
878 typedef enum OptixInstanceFlags
879 {
881     OPTIX_INSTANCE_FLAG_NONE = 0,
882
886     OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u « 0,
887
890     OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u « 1,
891
895     OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT = 1u « 2,
896
901     OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u « 3,
902
903
905     OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u « 4,
908     OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS = 1u « 5,
909
910 } OptixInstanceFlags;
911
915 typedef struct OptixInstance
916 {
918     float transform[12];
919
```

```
921     unsigned int instanceId;
922
926     unsigned int sbtOffset;
927
930     unsigned int visibilityMask;
931
933     unsigned int flags;
934
936     OptixTraversableHandle traversableHandle;
937
939     unsigned int pad[2];
940 } OptixInstance;
941
945 typedef enum OptixBuildFlags
946 {
948     OPTIX_BUILD_FLAG_NONE = 0,
949
952     OPTIX_BUILD_FLAG_ALLOW_UPDATE = 1u << 0,
953
954     OPTIX_BUILD_FLAG_ALLOW_COMPACTION = 1u << 1,
955
956     OPTIX_BUILD_FLAG_PREFER_FAST_TRACE = 1u << 2,
957
958     OPTIX_BUILD_FLAG_PREFER_FAST_BUILD = 1u << 3,
959
967     OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS = 1u << 4,
968
971     OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS = 1u << 5,
972
976     OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE = 1u << 6,
977
981     OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1u << 7,
982 } OptixBuildFlags;
983
984
986 typedef enum OptixOpacityMicromapFlags
987 {
988     OPTIX_OPACITY_MICROMAP_FLAG_NONE              = 0,
989     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 << 0,
990     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 << 1,
991 } OptixOpacityMicromapFlags;
992
994 typedef struct OptixOpacityMicromapDesc
995 {
997     unsigned int  byteOffset;
999     unsigned short subdivisionLevel;
1001     unsigned short format;
1002 } OptixOpacityMicromapDesc;
1003
1008 typedef struct OptixOpacityMicromapHistogramEntry
1009 {
1011     unsigned int               count;
1013     unsigned int               subdivisionLevel;
1015     OptixOpacityMicromapFormat format;
1016 } OptixOpacityMicromapHistogramEntry;
1017
1019 typedef struct OptixOpacityMicromapArrayBuildInput
1020 {
1022     OptixOpacityMicromapFlags flags;
1023
1025     CUdeviceptr inputBuffer;
1026
1028     CUdeviceptr perMicromapDescBuffer;
1029
1032     unsigned int perMicromapDescStrideInBytes;
1033
1035     unsigned int numMicromapHistogramEntries;
```

```
1038     const OptixOpacityMicromapHistogramEntry* micromapHistogramEntries;
1039 } OptixOpacityMicromapArrayBuildInput;
1040
1041
1043 typedef struct OptixMicromapBufferSizes
1044 {
1045     size_t outputSizeInBytes;
1046     size_t tempSizeInBytes;
1047 } OptixMicromapBufferSizes;
1048
1050 typedef struct OptixMicromapBuffers
1051 {
1053     CUdeviceptr output;
1055     size_t outputSizeInBytes;
1057     CUdeviceptr temp;
1059     size_t tempSizeInBytes;
1060 } OptixMicromapBuffers;
1061
1062
1063
1075 typedef enum OptixBuildOperation
1076 {
1078     OPTIX_BUILD_OPERATION_BUILD = 0x2161,
1080     OPTIX_BUILD_OPERATION_UPDATE = 0x2162,
1081 } OptixBuildOperation;
1082
1086 typedef enum OptixMotionFlags
1087 {
1088     OPTIX_MOTION_FLAG_NONE         = 0,
1089     OPTIX_MOTION_FLAG_START_VANISH = 1u « 0,
1090     OPTIX_MOTION_FLAG_END_VANISH   = 1u « 1
1091 } OptixMotionFlags;
1092
1097 typedef struct OptixMotionOptions
1098 {
1101     unsigned short numKeys;
1102
1104     unsigned short flags;
1105
1107     float timeBegin;
1108
1110     float timeEnd;
1111 } OptixMotionOptions;
1112
1116 typedef struct OptixAccelBuildOptions
1117 {
1119     unsigned int buildFlags;
1120
1127     OptixBuildOperation operation;
1128
1130     OptixMotionOptions motionOptions;
1131 } OptixAccelBuildOptions;
1132
1138 typedef struct OptixAccelBufferSizes
1139 {
1142     size_t outputSizeInBytes;
1143
1146     size_t tempSizeInBytes;
1147
1152     size_t tempUpdateSizeInBytes;
1153 } OptixAccelBufferSizes;
1154
1158 typedef enum OptixAccelPropertyType
1159 {
1161     OPTIX_PROPERTY_TYPE_COMPACTED_SIZE = 0x2181,
1162
1164     OPTIX_PROPERTY_TYPE_AABBS = 0x2182,
```

```
1165 } OptixAccelPropertyType;
1166
1170 typedef struct OptixAccelEmitDesc
1171 {
1173     CUdeviceptr result;
1174
1176     OptixAccelPropertyType type;
1177 } OptixAccelEmitDesc;
1178
1183 typedef struct OptixRelocationInfo
1184 {
1186     unsigned long long info[4];
1187 } OptixRelocationInfo;
1188
1194 typedef struct OptixStaticTransform
1195 {
1197     OptixTraversableHandle child;
1198
1200     unsigned int pad[2];
1201
1203     float transform[12];
1204
1207     float invTransform[12];
1208 } OptixStaticTransform;
1209
1234 typedef struct OptixMatrixMotionTransform
1235 {
1237     OptixTraversableHandle child;
1238
1241     OptixMotionOptions motionOptions;
1242
1244     unsigned int pad[3];
1245
1247     float transform[2][12];
1248 } OptixMatrixMotionTransform;
1249
1257 //      [ sx   a   b  pvx ]
1258 //  S = [  0  sy   c  pvy ]
1259 //      [  0   0  sz  pvz ]
1268 //      [  1  0  0 tx ]
1269 //  T = [  0  1  0 ty ]
1270 //      [  0  0  1 tz ]
1280 typedef struct OptixSRTData
1281 {
1284     float sx, a, b, pvx, sy, c, pvy, sz, pvz, qx, qy, qz, qw, tx, ty, tz;
1286 } OptixSRTData;
1287
1288 // TODO Define a static assert for C/pre-C++-11
1289 #if defined(__cplusplus) && __cplusplus >= 201103L
1290 static_assert(sizeof(OptixSRTData) == 16 * 4, "OptixSRTData has wrong size");
1291 #endif
1292
1317 typedef struct OptixSRTMotionTransform
1318 {
1320     OptixTraversableHandle child;
1321
1324     OptixMotionOptions motionOptions;
1325
1327     unsigned int pad[3];
1328
1330     OptixSRTData srtData[2];
1331 } OptixSRTMotionTransform;
1332
1333 // TODO Define a static assert for C/pre-C++-11
1334 #if defined(__cplusplus) && __cplusplus >= 201103L
1335 static_assert(sizeof(OptixSRTMotionTransform) == 8 + 12 + 12 + 2 * 16 * 4, "OptixSRTMotionTransform has
wrong size");
```

```
1336 #endif
1337
1341 typedef enum OptixTraversableType
1342 {
1344     OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM = 0x21C1,
1346     OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM = 0x21C2,
1348     OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM = 0x21C3,
1349 } OptixTraversableType;
1350
1354 typedef enum OptixPixelFormat
1355 {
1356     OPTIX_PIXEL_FORMAT_HALF2  = 0x2207,
1357     OPTIX_PIXEL_FORMAT_HALF3  = 0x2201,
1358     OPTIX_PIXEL_FORMAT_HALF4  = 0x2202,
1359     OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208,
1360     OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203,
1361     OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204,
1362     OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205,
1363     OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206,
1364     OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209,
1365 } OptixPixelFormat;
1366
1370 typedef struct OptixImage2D
1371 {
1373     CUdeviceptr data;
1375     unsigned int width;
1377     unsigned int height;
1379     unsigned int rowStrideInBytes;
1384     unsigned int pixelStrideInBytes;
1386     OptixPixelFormat format;
1387 } OptixImage2D;
1388
1392 typedef enum OptixDenoiserModelKind
1393 {
1395     OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322,
1396
1398     OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323,
1399
1401     OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324,
1402
1404     OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325,
1405
1407     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326,
1408
1410     OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327,
1411
1414     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328,
1415 } OptixDenoiserModelKind;
1416
1420 typedef struct OptixDenoiserOptions
1421 {
1422     // if nonzero, albedo image must be given in OptixDenoiserGuideLayer
1423     unsigned int guideAlbedo;
1424
1425     // if nonzero, normal image must be given in OptixDenoiserGuideLayer
1426     unsigned int guideNormal;
1427 } OptixDenoiserOptions;
1428
1432 typedef struct OptixDenoiserGuideLayer
1433 {
1434     // albedo/bsdf image
1435     OptixImage2D  albedo;
1436
1437     // normal vector image (2d or 3d pixel format)
1438     OptixImage2D  normal;
1439
1440     // 2d flow image, pixel flow from previous to current frame for each pixel
```

```
1441     OptixImage2D  flow;
1442
1443     OptixImage2D  previousOutputInternalGuideLayer;
1444     OptixImage2D  outputInternalGuideLayer;
1445 } OptixDenoiserGuideLayer;
1446
1450 typedef struct OptixDenoiserLayer
1451 {
1452     // input image (beauty or AOV)
1453     OptixImage2D  input;
1454
1455     // denoised output image from previous frame if temporal model kind selected
1456     OptixImage2D  previousOutput;
1457
1458     // denoised output for given input
1459     OptixImage2D  output;
1460 } OptixDenoiserLayer;
1461
1467 typedef enum OptixDenoiserAlphaMode
1468 {
1470     OPTIX_DENOISER_ALPHA_MODE_COPY = 0,
1471
1473     OPTIX_DENOISER_ALPHA_MODE_ALPHA_AS_AOV = 1,
1474
1477     OPTIX_DENOISER_ALPHA_MODE_FULL_DENOISE_PASS = 2
1478 } OptixDenoiserAlphaMode;
1479 typedef struct OptixDenoiserParams
1480 {
1482     OptixDenoiserAlphaMode denoiseAlpha;
1483
1487     CUdeviceptr  hdrIntensity;
1488
1493     float        blendFactor;
1494
1499     CUdeviceptr  hdrAverageColor;
1500
1505     unsigned int temporalModeUsePreviousLayers;
1506 } OptixDenoiserParams;
1507
1511 typedef struct OptixDenoiserSizes
1512 {
1514     size_t stateSizeInBytes;
1515
1518     size_t withOverlapScratchSizeInBytes;
1519
1522     size_t withoutOverlapScratchSizeInBytes;
1523
1525     unsigned int overlapWindowSizeInPixels;
1526
1529     size_t computeAverageColorSizeInBytes;
1530
1533     size_t computeIntensitySizeInBytes;
1534
1536     size_t internalGuideLayerPixelSizeInBytes;
1537 } OptixDenoiserSizes;
1538
1543 typedef enum OptixRayFlags
1544 {
1546     OPTIX_RAY_FLAG_NONE = 0u,
1547
1552     OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u « 0,
1553
1558     OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u « 1,
1559
1562     OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u « 2,
1563
1565     OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u « 3,
```

```
1566
1571     OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u « 4,
1572
1577     OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u « 5,
1578
1584     OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u « 6,
1585
1591     OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u « 7,
1592
1594     OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u « 10,
1595 } OptixRayFlags;
1596
1602 typedef enum OptixTransformType
1603 {
1604     OPTIX_TRANSFORM_TYPE_NONE                     = 0,
1605     OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM         = 1,
1606     OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM  = 2,
1607     OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM     = 3,
1608     OPTIX_TRANSFORM_TYPE_INSTANCE                 = 4,
1609 } OptixTransformType;
1610
1613 typedef enum OptixTraversableGraphFlags
1614 {
1617     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0,
1618
1622     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u « 0,
1623
1628     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u « 1,
1629 } OptixTraversableGraphFlags;
1630
1634 typedef enum OptixCompileOptimizationLevel
1635 {
1637     OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0,
1639     OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340,
1641     OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341,
1643     OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342,
1645     OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343,
1646 } OptixCompileOptimizationLevel;
1647
1651 typedef enum OptixCompileDebugLevel
1652 {
1654     OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT  = 0,
1656     OPTIX_COMPILE_DEBUG_LEVEL_NONE     = 0x2350,
1659     OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL  = 0x2351,
1661     OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353,
1663     OPTIX_COMPILE_DEBUG_LEVEL_FULL     = 0x2352,
1664 } OptixCompileDebugLevel;
1665
1669 typedef enum OptixModuleCompileState
1670 {
1672     OPTIX_MODULE_COMPILE_STATE_NOT_STARTED       = 0x2360,
1673
1675     OPTIX_MODULE_COMPILE_STATE_STARTED           = 0x2361,
1676
1678     OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE = 0x2362,
1679
1681     OPTIX_MODULE_COMPILE_STATE_FAILED            = 0x2363,
1682
1684     OPTIX_MODULE_COMPILE_STATE_COMPLETED         = 0x2364,
1685 } OptixModuleCompileState;
1686
1687
1688
1721 typedef struct OptixModuleCompileBoundValueEntry {
1722     size_t pipelineParamOffsetInBytes;
1723     size_t sizeInBytes;
1724     const void* boundValuePtr;
```

```
1725     const char* annotation; // optional string to display, set to 0 if unused.  If unused,
1726                             // OptiX will report the annotation as "No annotation"
1727 } OptixModuleCompileBoundValueEntry;
1728
1730 typedef enum OptixPayloadTypeID {
1731     OPTIX_PAYLOAD_TYPE_DEFAULT = 0,
1732     OPTIX_PAYLOAD_TYPE_ID_0 = (1 « 0u),
1733     OPTIX_PAYLOAD_TYPE_ID_1 = (1 « 1u),
1734     OPTIX_PAYLOAD_TYPE_ID_2 = (1 « 2u),
1735     OPTIX_PAYLOAD_TYPE_ID_3 = (1 « 3u),
1736     OPTIX_PAYLOAD_TYPE_ID_4 = (1 « 4u),
1737     OPTIX_PAYLOAD_TYPE_ID_5 = (1 « 5u),
1738     OPTIX_PAYLOAD_TYPE_ID_6 = (1 « 6u),
1739     OPTIX_PAYLOAD_TYPE_ID_7 = (1 « 7u)
1740 } OptixPayloadTypeID;
1741
1755 typedef enum OptixPayloadSemantics
1756 {
1757     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE       = 0,
1758     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ       = 1u « 0,
1759     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE      = 2u « 0,
1760     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u « 0,
1761
1762     OPTIX_PAYLOAD_SEMANTICS_CH_NONE                 = 0,
1763     OPTIX_PAYLOAD_SEMANTICS_CH_READ                 = 1u « 2,
1764     OPTIX_PAYLOAD_SEMANTICS_CH_WRITE                = 2u « 2,
1765     OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE           = 3u « 2,
1766
1767     OPTIX_PAYLOAD_SEMANTICS_MS_NONE                 = 0,
1768     OPTIX_PAYLOAD_SEMANTICS_MS_READ                 = 1u « 4,
1769     OPTIX_PAYLOAD_SEMANTICS_MS_WRITE                = 2u « 4,
1770     OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE           = 3u « 4,
1771
1772     OPTIX_PAYLOAD_SEMANTICS_AH_NONE                 = 0,
1773     OPTIX_PAYLOAD_SEMANTICS_AH_READ                 = 1u « 6,
1774     OPTIX_PAYLOAD_SEMANTICS_AH_WRITE                = 2u « 6,
1775     OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE           = 3u « 6,
1776
1777     OPTIX_PAYLOAD_SEMANTICS_IS_NONE                 = 0,
1778     OPTIX_PAYLOAD_SEMANTICS_IS_READ                 = 1u « 8,
1779     OPTIX_PAYLOAD_SEMANTICS_IS_WRITE                = 2u « 8,
1780     OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE           = 3u « 8,
1781 } OptixPayloadSemantics;
1782
1784 typedef struct OptixPayloadType
1785 {
1787     unsigned int numPayloadValues;
1788
1790     const unsigned int *payloadSemantics;
1791 } OptixPayloadType;
1792
1796 typedef struct OptixModuleCompileOptions
1797 {
1800     int maxRegisterCount;
1801
1803     OptixCompileOptimizationLevel optLevel;
1804
1806     OptixCompileDebugLevel debugLevel;
1807
1809     const OptixModuleCompileBoundValueEntry* boundValues;
1810
1812     unsigned int numBoundValues;
1813
1816     unsigned int numPayloadTypes;
1817
1819     OptixPayloadType *payloadTypes;
1820
```

```
1821 } OptixModuleCompileOptions;
1822
1824 typedef enum OptixProgramGroupKind
1825 {
1828     OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421,
1829
1832     OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422,
1833
1836     OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423,
1837
1840     OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424,
1841
1844     OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425
1845 } OptixProgramGroupKind;
1846
1848 typedef enum OptixProgramGroupFlags
1849 {
1851     OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0
1852 } OptixProgramGroupFlags;
1853
1860 typedef struct OptixProgramGroupSingleModule
1861 {
1863     OptixModule module;
1865     const char* entryFunctionName;
1866 } OptixProgramGroupSingleModule;
1867
1873 typedef struct OptixProgramGroupHitgroup
1874 {
1876     OptixModule moduleCH;
1878     const char* entryFunctionNameCH;
1880     OptixModule moduleAH;
1882     const char* entryFunctionNameAH;
1884     OptixModule moduleIS;
1886     const char* entryFunctionNameIS;
1887 } OptixProgramGroupHitgroup;
1888
1894 typedef struct OptixProgramGroupCallables
1895 {
1897     OptixModule moduleDC;
1899     const char* entryFunctionNameDC;
1901     OptixModule moduleCC;
1903     const char* entryFunctionNameCC;
1904 } OptixProgramGroupCallables;
1905
1907 typedef struct OptixProgramGroupDesc
1908 {
1910     OptixProgramGroupKind kind;
1911
1913     unsigned int flags;
1914
1915     union
1916     {
1918         OptixProgramGroupSingleModule raygen;
1920         OptixProgramGroupSingleModule miss;
1922         OptixProgramGroupSingleModule exception;
1924         OptixProgramGroupCallables callables;
1926         OptixProgramGroupHitgroup hitgroup;
1927     };
1928 } OptixProgramGroupDesc;
1929
1933 typedef struct OptixProgramGroupOptions
1934 {
1947     OptixPayloadType* payloadType;
1948 } OptixProgramGroupOptions;
1949
1951 typedef enum OptixExceptionCodes
1952 {
```

```
1955     OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1,
1956
1959     OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2,
1960
1965     OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED = -3,
1966
1972     OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE = -5,
1973
1978     OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT = -6,
1979
1984     //      sbt-index (See optixGetExceptionInvalidSbtOffset),
1985     //      sbt-instance-offset (See OptixInstance::sbtOffset),
1996     OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT = -7,
1997
2000     OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE = -8,
2001
2006     OPTIX_EXCEPTION_CODE_INVALID_RAY = -9,
2007
2013     OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH = -10,
2014
2016     OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH = -11,
2017
2022     OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT = -12,
2023
2026     OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD = -13,
2027
2030     OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD = -14,
2031
2038     OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS = -15,
2039
2042     OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0 = -16,
2043     OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_1 = -17,
2044     OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2 = -18,
2045
2047     OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS = -32,
2048
2050     OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH = -33,
2051 } OptixExceptionCodes;
2052
2056 typedef enum OptixExceptionFlags
2057 {
2059     OPTIX_EXCEPTION_FLAG_NONE = 0,
2060
2062     OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u « 0,
2063
2065     OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u « 1,
2066
2069     OPTIX_EXCEPTION_FLAG_USER = 1u « 2,
2070
2072     OPTIX_EXCEPTION_FLAG_DEBUG = 1u « 3
2073 } OptixExceptionFlags;
2074
2080 typedef struct OptixPipelineCompileOptions
2081 {
2083     int usesMotionBlur;
2084
2086     unsigned int traversableGraphFlags;
2087
2090     int numPayloadValues;
2091
2094     int numAttributeValues;
2095
2097     unsigned int exceptionFlags;
2098
2102     const char* pipelineLaunchParamsVariableName;
2103
2106     unsigned int usesPrimitiveTypeFlags;
```

```
2107
2109     int allowOpacityMicromaps;
2110 } OptixPipelineCompileOptions;
2111
2115 typedef struct OptixPipelineLinkOptions
2116 {
2119     unsigned int maxTraceDepth;
2120
2122     OptixCompileDebugLevel debugLevel;
2123 } OptixPipelineLinkOptions;
2124
2128 typedef struct OptixShaderBindingTable
2129 {
2132     CUdeviceptr raygenRecord;
2133
2136     CUdeviceptr exceptionRecord;
2137
2141     CUdeviceptr  missRecordBase;
2142     unsigned int missRecordStrideInBytes;
2143     unsigned int missRecordCount;
2145
2149     CUdeviceptr  hitgroupRecordBase;
2150     unsigned int hitgroupRecordStrideInBytes;
2151     unsigned int hitgroupRecordCount;
2153
2158     CUdeviceptr  callablesRecordBase;
2159     unsigned int callablesRecordStrideInBytes;
2160     unsigned int callablesRecordCount;
2162
2163 } OptixShaderBindingTable;
2164
2168 typedef struct OptixStackSizes
2169 {
2171     unsigned int cssRG;
2173     unsigned int cssMS;
2175     unsigned int cssCH;
2177     unsigned int cssAH;
2179     unsigned int cssIS;
2181     unsigned int cssCC;
2183     unsigned int dssDC;
2184
2185 } OptixStackSizes;
2186
2188 typedef enum OptixQueryFunctionTableOptions
2189 {
2191     OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0
2192
2193 } OptixQueryFunctionTableOptions;
2194
2196 typedef OptixResult(OptixQueryFunctionTable_t)(int          abiId,
2197                                                unsigned int numOptions,
2198                                                OptixQueryFunctionTableOptions* /*optionKeys*/,
2199                                                const void** /*optionValues*/,
2200                                                void*  functionTable,
2201                                                size_t sizeOfTable);
2202
2207 typedef struct OptixBuiltinISOptions
2208 {
2209     OptixPrimitiveType        builtinISModuleType;
2211     int                       usesMotionBlur;
2213     unsigned int              buildFlags;
2215     unsigned int              curveEndcapFlags;
2216 } OptixBuiltinISOptions;
2217
2218 #if defined(__CUDACC__)
2223 typedef struct OptixInvalidRayExceptionDetails
2224 {
```

```
2225     float3 origin;
2226     float3 direction;
2227     float  tmin;
2228     float  tmax;
2229     float  time;
2230 } OptixInvalidRayExceptionDetails;
2231
2238 typedef struct OptixParameterMismatchExceptionDetails
2239 {
2241     unsigned int expectedParameterCount;
2243     unsigned int passedArgumentCount;
2245     unsigned int sbtIndex;
2247     char*        callableName;
2248 } OptixParameterMismatchExceptionDetails;
2249 #endif
2250
2251   // end group optix_types
2253
2254 #endif  // __optix_optix_7_types_h__
```

8.15 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)

8.15.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.16 optix_denoiser_tiling.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
3 *
4 * Redistribution and use in source and binary forms, with or without
5 * modification, are permitted provided that the following conditions
6 * are met:
7 *  * Redistributions of source code must retain the above copyright
8 *    notice, this list of conditions and the following disclaimer.
9 *  * Redistributions in binary form must reproduce the above copyright
10 *    notice, this list of conditions and the following disclaimer in the
11 *    documentation and/or other materials provided with the distribution.
```

```
12 *  * Neither the name of NVIDIA CORPORATION nor the names of its
13 *    contributors may be used to endorse or promote products derived
14 *    from this software without specific prior written permission.
15 *
16 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS “AS IS” AND ANY
17 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
18 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
19 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
20 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
21 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
22 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
23 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
24 * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
25 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
26 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
27 */
28
32
33 #ifndef optix_denoiser_tiling_h
34 #define optix_denoiser_tiling_h
35
36 #include <optix.h>
37
38 #include <algorithm>
39 #include <vector>
40
41 #ifdef __cplusplus
42 extern "C" {
43 #endif
44
53 struct OptixUtilDenoiserImageTile
54 {
55     // input tile image
56     OptixImage2D input;
57
58     // output tile image
59     OptixImage2D output;
60
61     // overlap offsets, parameters for #optixUtilDenoiserInvoke
62     unsigned int inputOffsetX;
63     unsigned int inputOffsetY;
64 };
65
72 inline OptixResult optixUtilGetPixelStride(const OptixImage2D& image, unsigned int& pixelStrideInBytes)
73 {
74     pixelStrideInBytes = image.pixelStrideInBytes;
75     if(pixelStrideInBytes == 0)
76     {
77         switch(image.format)
78         {
79             case OPTIX_PIXEL_FORMAT_HALF2:
80                 pixelStrideInBytes = 2 * sizeof(short);
81                 break;
82             case OPTIX_PIXEL_FORMAT_HALF3:
83                 pixelStrideInBytes = 3 * sizeof(short);
84                 break;
85             case OPTIX_PIXEL_FORMAT_HALF4:
86                 pixelStrideInBytes = 4 * sizeof(short);
87                 break;
88             case OPTIX_PIXEL_FORMAT_FLOAT2:
89                 pixelStrideInBytes = 2 * sizeof(float);
90                 break;
91             case OPTIX_PIXEL_FORMAT_FLOAT3:
92                 pixelStrideInBytes = 3 * sizeof(float);
93                 break;
94             case OPTIX_PIXEL_FORMAT_FLOAT4:
95                 pixelStrideInBytes = 4 * sizeof(float);
```

```
96                 break;
97             case OPTIX_PIXEL_FORMAT_UCHAR3:
98                 pixelStrideInBytes = 3 * sizeof(char);
99                 break;
100             case OPTIX_PIXEL_FORMAT_UCHAR4:
101                 pixelStrideInBytes = 4 * sizeof(char);
102                 break;
103             case OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER:
104                 return OPTIX_ERROR_INVALID_VALUE;
105                 break;
106         }
107     }
108     return OPTIX_SUCCESS;
109 }
110
120 inline OptixResult optixUtilDenoiserSplitImage(
121                                                 const OptixImage2D&                    input,
122                                                 const OptixImage2D&                    output,
123                                           unsigned int                         overlapWindowSizeInPixels,
124                                                 unsigned int                           tileWidth,
125                                                 unsigned int                           tileHeight,
126                                                 std::vector<OptixUtilDenoiserImageTile>&    tiles)
127 {
128     if(tileWidth == 0 || tileHeight == 0)
129         return OPTIX_ERROR_INVALID_VALUE;
130
131     unsigned int inPixelStride, outPixelStride;
132     if(const OptixResult res = optixUtilGetPixelStride(input, inPixelStride))
133         return res;
134     if(const OptixResult res = optixUtilGetPixelStride(output, outPixelStride))
135         return res;
136
137     int inp_w = std::min(tileWidth + 2 * overlapWindowSizeInPixels, input.width);
138     int inp_h = std::min(tileHeight + 2 * overlapWindowSizeInPixels, input.height);
139     int inp_y = 0, copied_y = 0;
140
141     int upscaleX = output.width / input.width;
142     int upscaleY = output.height / input.height;
143
144     do
145     {
146         int inputOffsetY = inp_y == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_h -
((int)input.height - inp_y));
147         int copy_y       = inp_y == 0 ? std::min(input.height, tileHeight + overlapWindowSizeInPixels) :
148                                         std::min(tileHeight, input.height - copied_y);
149
150         int inp_x = 0, copied_x = 0;
151         do
152         {
153             int inputOffsetX = inp_x == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_w -
((int)input.width - inp_x));
154             int copy_x = inp_x == 0 ? std::min(input.width, tileWidth + overlapWindowSizeInPixels) :
155                                       std::min(tileWidth, input.width - copied_x);
156
157             OptixUtilDenoiserImageTile tile;
158             tile.input.data              = input.data + (size_t)(inp_y - inputOffsetY) *
input.rowStrideInBytes
159                                            + (size_t)(inp_x - inputOffsetX) * inPixelStride;
160             tile.input.width             = inp_w;
161             tile.input.height            = inp_h;
162             tile.input.rowStrideInBytes   = input.rowStrideInBytes;
163             tile.input.pixelStrideInBytes = input.pixelStrideInBytes;
164             tile.input.format            = input.format;
165
166             tile.output.data              = output.data + (size_t)(upscaleY * inp_y) *
output.rowStrideInBytes
167                                             + (size_t)(upscaleX * inp_x) * outPixelStride;
```

```
168             tile.output.width              = upscaleX * copy_x;
169             tile.output.height             = upscaleY * copy_y;
170             tile.output.rowStrideInBytes   = output.rowStrideInBytes;
171             tile.output.pixelStrideInBytes = output.pixelStrideInBytes;
172             tile.output.format             = output.format;
173 
174             tile.inputOffsetX = inputOffsetX;
175             tile.inputOffsetY = inputOffsetY;
176 
177             tiles.push_back(tile);
178 
179             inp_x += inp_x == 0 ? tileWidth + overlapWindowSizeInPixels : tileWidth;
180             copied_x += copy_x;
181         } while(inp_x < static_cast<int>(input.width));
182 
183         inp_y += inp_y == 0 ? tileHeight + overlapWindowSizeInPixels : tileHeight;
184         copied_y += copy_y;
185     } while(inp_y < static_cast<int>(input.height));
186 
187     return OPTIX_SUCCESS;
188 }
189 
193 
200 
216 inline OptixResult optixUtilDenoiserInvokeTiled(
217                                                 OptixDenoiser                   denoiser,
218                                                 CUstream                        stream,
219                                                 const OptixDenoiserParams*      params,
220                                                 CUdeviceptr                     denoiserState,
221                                                 size_t                          denoiserStateSizeInBytes,
222                                                 const OptixDenoiserGuideLayer*  guideLayer,
223                                                 const OptixDenoiserLayer*       layers,
224                                                 unsigned int                    numLayers,
225                                                 CUdeviceptr                     scratch,
226                                                 size_t                          scratchSizeInBytes,
227                                                 unsigned int                    overlapWindowSizeInPixels,
228                                                 unsigned int                    tileWidth,
229                                                 unsigned int                    tileHeight)
230 {
231     if(!guideLayer || !layers)
232         return OPTIX_ERROR_INVALID_VALUE;
233 
234     const unsigned int upscale = numLayers > 0 && layers[0].previousOutput.width == 2 *
layers[0].input.width ? 2 : 1;
235 
236     std::vector<std::vector<OptixUtilDenoiserImageTile» tiles(numLayers);
237     std::vector<std::vector<OptixUtilDenoiserImageTile» prevTiles(numLayers);
238     for(unsigned int l = 0; l < numLayers; l++)
239     {
240         if(const OptixResult res = optixUtilDenoiserSplitImage(layers[l].input, layers[l].output,
241                                                                overlapWindowSizeInPixels,
242                                                                tileWidth, tileHeight, tiles[l]))
243             return res;
244 
245         if(layers[l].previousOutput.data)
246         {
247             OptixImage2D dummyOutput = layers[l].previousOutput;
248             if(const OptixResult res = optixUtilDenoiserSplitImage(layers[l].previousOutput, dummyOutput,
249                                                                    upscale * overlapWindowSizeInPixels,
250                                                                upscale * tileWidth, upscale * tileHeight,
prevTiles[l]))
251                 return res;
252         }
253     }
254 
255     std::vector<OptixUtilDenoiserImageTile> albedoTiles;
256     if(guideLayer->albedo.data)
```

```
257     {
258         OptixImage2D dummyOutput = guideLayer->albedo;
259         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->albedo, dummyOutput,
260                                                                overlapWindowSizeInPixels,
261                                                                tileWidth, tileHeight, albedoTiles))
262             return res;
263     }
264
265     std::vector<OptixUtilDenoiserImageTile> normalTiles;
266     if(guideLayer->normal.data)
267     {
268         OptixImage2D dummyOutput = guideLayer->normal;
269         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->normal, dummyOutput,
270                                                                overlapWindowSizeInPixels,
271                                                                tileWidth, tileHeight, normalTiles))
272             return res;
273     }
274     std::vector<OptixUtilDenoiserImageTile> flowTiles;
275     if(guideLayer->flow.data)
276     {
277         OptixImage2D dummyOutput = guideLayer->flow;
278         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->flow, dummyOutput,
279                                                                overlapWindowSizeInPixels,
280                                                                tileWidth, tileHeight, flowTiles))
281             return res;
282     }
283
284     std::vector<OptixUtilDenoiserImageTile> internalGuideLayerTiles;
285     if(guideLayer->previousOutputInternalGuideLayer.data && guideLayer->outputInternalGuideLayer.data)
286     {
287         if(const OptixResult res =
optixUtilDenoiserSplitImage(guideLayer->previousOutputInternalGuideLayer,
288                                                                guideLayer->outputInternalGuideLayer,
289                                                                upscale * overlapWindowSizeInPixels,
290                                                              upscale * tileWidth, upscale * tileHeight,
internalGuideLayerTiles))
291             return res;
292     }
293
294     for(size_t t = 0; t < tiles[0].size(); t++)
295     {
296         std::vector<OptixDenoiserLayer> tlayers;
297         for(unsigned int l = 0; l < numLayers; l++)
298         {
299             OptixDenoiserLayer layer = {};
300             layer.input  = (tiles[l])[t].input;
301             layer.output = (tiles[l])[t].output;
302             if(layers[l].previousOutput.data)
303                 layer.previousOutput = (prevTiles[l])[t].input;
304             tlayers.push_back(layer);
305         }
306
307         OptixDenoiserGuideLayer gl = {};
308         if(guideLayer->albedo.data)
309             gl.albedo = albedoTiles[t].input;
310
311         if(guideLayer->normal.data)
312             gl.normal = normalTiles[t].input;
313
314         if(guideLayer->flow.data)
315             gl.flow = flowTiles[t].input;
316
317         if(guideLayer->previousOutputInternalGuideLayer.data)
318             gl.previousOutputInternalGuideLayer = internalGuideLayerTiles[t].input;
319
320         if(guideLayer->outputInternalGuideLayer.data)
321             gl.outputInternalGuideLayer = internalGuideLayerTiles[t].output;
```

```
322
323         if(const OptixResult res =
324                 optixDenoiserInvoke(denoiser, stream, params, denoiserState, denoiserStateSizeInBytes,
325                                     &gl, &tlayers[0], numLayers,
326                                     (tiles[0])[t].inputOffsetX, (tiles[0])[t].inputOffsetY,
327                                     scratch, scratchSizeInBytes))
328             return res;
329     }
330     return OPTIX_SUCCESS;
331 }
332   // end group optix_utilities
334
335 #ifdef __cplusplus
336 }
337 #endif
338
339 #endif  // __optix_optix_stack_size_h__
```

8.17 optix_device.h File Reference

Macros

- #define __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_DEVICE_H__

8.17.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Host/Device side

8.17.2 Macro Definition Documentation

8.17.2.1 __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_DEVICE_H__

```
#define __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_DEVICE_H__
```

8.18 optix_device.h

Go to the documentation of this file.

```
1
2 /*
3 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
4 *
5 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
6 * rights in and to this software, related documentation and any modifications thereto.
7 * Any use, reproduction, disclosure or distribution of this software and related
8 * documentation without an express license agreement from NVIDIA Corporation is strictly
9 * prohibited.
10 *
11 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
12 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
13 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
14 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
15 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
16 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
17 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
18 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
19 * SUCH DAMAGES
```

```
20 */
21
30 /**************************************************************************\
31 * optix_cuda.h
32 *
33 * This file provides the nvcc interface for generating PTX that the OptiX is
34 * capable of parsing and weaving into the final kernel.  This is included by
35 * optix.h automatically if compiling device code.  It can be included explicitly
36 * in host code if desired.
37 *
38 \**************************************************************************/
39 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
40 #  define __OPTIX_INCLUDE_INTERNAL_HEADERS__
41 #  define __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_DEVICE_H__
42 #endif
43 #include "optix_7_device.h"
44 #if defined(__UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_DEVICE_H__)
45 #  undef __OPTIX_INCLUDE_INTERNAL_HEADERS__
46 #  undef __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_DEVICE_H__
47 #endif
```

8.19 optix_function_table.h File Reference

Classes

- struct OptixFunctionTable

Macros

- #define OPTIX_ABI_VERSION 68

Typedefs

- typedef struct OptixFunctionTable OptixFunctionTable

8.19.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.19.2 Macro Definition Documentation

8.19.2.1 OPTIX_ABI_VERSION

```
#define OPTIX_ABI_VERSION 68
```

The OptiX ABI version.

8.20 optix_function_table.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
3 *
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly
8 * prohibited.
```

```
 9 *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
24
25 #ifndef __optix_optix_function_table_h__
26 #define __optix_optix_function_table_h__
27
29 #define OPTIX_ABI_VERSION 68
30
31 #ifndef OPTIX_DEFINE_ABI_VERSION_ONLY
32
33 #include "optix_types.h"
34
35 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
36 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver types must be defined through other
37 // means before including optix headers.
38 #include <cuda.h>
39 #endif
40
41 #ifdef __cplusplus
42 extern "C" {
43 #endif
44
47
55 typedef struct OptixFunctionTable
56 {
58     //@ {
59
61     const char* (*optixGetErrorName)(OptixResult result);
62
64     const char* (*optixGetErrorString)(OptixResult result);
65
66     //@ }
68     //@ {
69
71     OptixResult (*optixDeviceContextCreate)(CUcontext fromContext, const OptixDeviceContextOptions*
options, OptixDeviceContext* context);
72
74     OptixResult (*optixDeviceContextDestroy)(OptixDeviceContext context);
75
77     OptixResult (*optixDeviceContextGetProperty)(OptixDeviceContext context, OptixDeviceProperty
property, void* value, size_t sizeInBytes);
78
80     OptixResult (*optixDeviceContextSetLogCallback)(OptixDeviceContext context,
81                                                     OptixLogCallback   callbackFunction,
82                                                     void*              callbackData,
83                                                     unsigned int       callbackLevel);
84
86     OptixResult (*optixDeviceContextSetCacheEnabled)(OptixDeviceContext context, int enabled);
87
89     OptixResult (*optixDeviceContextSetCacheLocation)(OptixDeviceContext context, const char* location);
90
92     OptixResult (*optixDeviceContextSetCacheDatabaseSizes)(OptixDeviceContext context, size_t
lowWaterMark, size_t highWaterMark);
93
95     OptixResult (*optixDeviceContextGetCacheEnabled)(OptixDeviceContext context, int* enabled);
96
98     OptixResult (*optixDeviceContextGetCacheLocation)(OptixDeviceContext context, char* location, size_t
```

```
locationSize);
99
101     OptixResult (*optixDeviceContextGetCacheDatabaseSizes)(OptixDeviceContext context, size_t*
lowWaterMark, size_t* highWaterMark);
102
103     //@ }
105     //@ {
106
108     OptixResult (*optixModuleCreateFromPTX)(OptixDeviceContext                 context,
109                                             const OptixModuleCompileOptions*   moduleCompileOptions,
110                                             const OptixPipelineCompileOptions* pipelineCompileOptions,
111                                             const char*                        PTX,
112                                             size_t                             PTXsize,
113                                             char*                              logString,
114                                             size_t*                            logStringSize,
115                                             OptixModule*                       module);
116
118     OptixResult (*optixModuleCreateFromPTXWithTasks)(OptixDeviceContext                 context,
119                                                      const OptixModuleCompileOptions*
moduleCompileOptions,
120                                                      const OptixPipelineCompileOptions*
pipelineCompileOptions,
121                                                      const char*                        PTX,
122                                                      size_t                             PTXsize,
123                                                      char*                              logString,
124                                                      size_t*                            logStringSize,
125                                                      OptixModule*                       module,
126                                                      OptixTask*                         firstTask);
127
129     OptixResult (*optixModuleGetCompilationState)(OptixModule module, OptixModuleCompileState* state);
130
132     OptixResult (*optixModuleDestroy)(OptixModule module);
133
135     OptixResult(*optixBuiltinISModuleGet)(OptixDeviceContext                 context,
136                                           const OptixModuleCompileOptions*   moduleCompileOptions,
137                                           const OptixPipelineCompileOptions* pipelineCompileOptions,
138                                           const OptixBuiltinISOptions*       builtinISOptions,
139                                           OptixModule*                       builtinModule);
140
141     //@ }
143     //@ {
144
146     OptixResult (*optixTaskExecute)(OptixTask     task,
147                                     OptixTask*    additionalTasks,
148                                     unsigned int  maxNumAdditionalTasks,
149                                     unsigned int* numAdditionalTasksCreated);
150     //@ }
152     //@ {
153
155     OptixResult (*optixProgramGroupCreate)(OptixDeviceContext              context,
156                                            const OptixProgramGroupDesc*    programDescriptions,
157                                            unsigned int                    numProgramGroups,
158                                            const OptixProgramGroupOptions* options,
159                                            char*                           logString,
160                                            size_t*                         logStringSize,
161                                            OptixProgramGroup*              programGroups);
162
164     OptixResult (*optixProgramGroupDestroy)(OptixProgramGroup programGroup);
165
167     OptixResult (*optixProgramGroupGetStackSize)(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes);
168
169     //@ }
171     //@ {
172
174     OptixResult (*optixPipelineCreate)(OptixDeviceContext                 context,
175                                        const OptixPipelineCompileOptions* pipelineCompileOptions,
```

```
176                                         const OptixPipelineLinkOptions*    pipelineLinkOptions,
177                                         const OptixProgramGroup*           programGroups,
178                                         unsigned int                       numProgramGroups,
179                                         char*                              logString,
180                                         size_t*                            logStringSize,
181                                         OptixPipeline*                     pipeline);
182
184     OptixResult (*optixPipelineDestroy)(OptixPipeline pipeline);
185
187     OptixResult (*optixPipelineSetStackSize)(OptixPipeline pipeline,
188                                              unsigned int  directCallableStackSizeFromTraversal,
189                                              unsigned int  directCallableStackSizeFromState,
190                                              unsigned int  continuationStackSize,
191                                              unsigned int  maxTraversableGraphDepth);
192
193     //@ }
195     //@ {
196
198     OptixResult (*optixAccelComputeMemoryUsage)(OptixDeviceContext            context,
199                                                 const OptixAccelBuildOptions* accelOptions,
200                                                 const OptixBuildInput*        buildInputs,
201                                                 unsigned int                  numBuildInputs,
202                                                 OptixAccelBufferSizes*        bufferSizes);
203
205     OptixResult (*optixAccelBuild)(OptixDeviceContext            context,
206                                    CUstream                      stream,
207                                    const OptixAccelBuildOptions* accelOptions,
208                                    const OptixBuildInput*        buildInputs,
209                                    unsigned int                  numBuildInputs,
210                                    CUdeviceptr                   tempBuffer,
211                                    size_t                        tempBufferSizeInBytes,
212                                    CUdeviceptr                   outputBuffer,
213                                    size_t                        outputBufferSizeInBytes,
214                                    OptixTraversableHandle*       outputHandle,
215                                    const OptixAccelEmitDesc*     emittedProperties,
216                                    unsigned int                  numEmittedProperties);
217
219     OptixResult (*optixAccelGetRelocationInfo)(OptixDeviceContext context, OptixTraversableHandle
handle, OptixRelocationInfo* info);
220
221
223     OptixResult (*optixCheckRelocationCompatibility)(OptixDeviceContext        context,
224                                                      const OptixRelocationInfo* info,
225                                                      int*                       compatible);
226
228     OptixResult (*optixAccelRelocate)(OptixDeviceContext         context,
229                                       CUstream                   stream,
230                                       const OptixRelocationInfo* info,
231                                       const OptixRelocateInput*  relocateInputs,
232                                       size_t                     numRelocateInputs,
233                                       CUdeviceptr                targetAccel,
234                                       size_t                     targetAccelSizeInBytes,
235                                       OptixTraversableHandle*    targetHandle);
236
237
239     OptixResult (*optixAccelCompact)(OptixDeviceContext      context,
240                                      CUstream                stream,
241                                      OptixTraversableHandle  inputHandle,
242                                      CUdeviceptr             outputBuffer,
243                                      size_t                  outputBufferSizeInBytes,
244                                      OptixTraversableHandle* outputHandle);
245
247     OptixResult (*optixConvertPointerToTraversableHandle)(OptixDeviceContext      onDevice,
248                                                           CUdeviceptr             pointer,
249                                                           OptixTraversableType    traversableType,
250                                                           OptixTraversableHandle* traversableHandle);
251
```

```
253     OptixResult (*optixOpacityMicromapArrayComputeMemoryUsage)(OptixDeviceContext
context,
254                                                                const OptixOpacityMicromapArrayBuildInput*
buildInput,
255                                                                   OptixMicromapBufferSizes*
bufferSizes);
256
258     OptixResult (*optixOpacityMicromapArrayBuild)(OptixDeviceContext                       context,
259                                                   CUstream                                    stream,
260                                                const OptixOpacityMicromapArrayBuildInput* buildInput,
261                                                   const OptixMicromapBuffers*                buffers);
262
264     OptixResult (*optixOpacityMicromapArrayGetRelocationInfo)(OptixDeviceContext   context,
265                                                               CUdeviceptr          opacityMicromapArray,
266                                                                 OptixRelocationInfo* info);
267
269     OptixResult (*optixOpacityMicromapArrayRelocate)(OptixDeviceContext         context,
270                                                      CUstream                      stream,
271                                                      const OptixRelocationInfo* info,
272                                                   CUdeviceptr               targetOpacityMicromapArray,
273                                                      size_t
targetOpacityMicromapArraySizeInBytes);
274
275     void (*reserved1)(void);
276     void (*reserved2)(void);
277
278     //@ }
280     //@ {
281
283     OptixResult (*optixSbtRecordPackHeader)(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer);
284
286     OptixResult (*optixLaunch)(OptixPipeline                 pipeline,
287                                CUstream                      stream,
288                                CUdeviceptr                   pipelineParams,
289                                size_t                        pipelineParamsSize,
290                                const OptixShaderBindingTable* sbt,
291                                unsigned int                  width,
292                                unsigned int                  height,
293                                unsigned int                  depth);
294
295     //@ }
297     //@ {
298
300     OptixResult (*optixDenoiserCreate)(OptixDeviceContext context, OptixDenoiserModelKind modelKind,
const OptixDenoiserOptions* options, OptixDenoiser* returnHandle);
301
303     OptixResult (*optixDenoiserDestroy)(OptixDenoiser handle);
304
306     OptixResult (*optixDenoiserComputeMemoryResources)(const OptixDenoiser handle,
307                                                        unsigned int        maximumInputWidth,
308                                                        unsigned int        maximumInputHeight,
309                                                        OptixDenoiserSizes* returnSizes);
310
312     OptixResult (*optixDenoiserSetup)(OptixDenoiser denoiser,
313                                       CUstream      stream,
314                                       unsigned int  inputWidth,
315                                       unsigned int  inputHeight,
316                                       CUdeviceptr   state,
317                                       size_t        stateSizeInBytes,
318                                       CUdeviceptr   scratch,
319                                       size_t        scratchSizeInBytes);
320
322     OptixResult (*optixDenoiserInvoke)(OptixDenoiser                   denoiser,
323                                        CUstream                           stream,
324                                        const OptixDenoiserParams*         params,
325                                        CUdeviceptr                        denoiserState,
```

```
326                                             size_t                          denoiserStateSizeInBytes,
327                                             const OptixDenoiserGuideLayer * guideLayer,
328                                             const OptixDenoiserLayer *      layers,
329                                             unsigned int                    numLayers,
330                                             unsigned int                    inputOffsetX,
331                                             unsigned int                    inputOffsetY,
332                                             CUdeviceptr                     scratch,
333                                             size_t                          scratchSizeInBytes);
334
336     OptixResult (*optixDenoiserComputeIntensity)(OptixDenoiser       handle,
337                                                  CUstream            stream,
338                                                  const OptixImage2D* inputImage,
339                                                  CUdeviceptr         outputIntensity,
340                                                  CUdeviceptr         scratch,
341                                                  size_t              scratchSizeInBytes);
342
344     OptixResult (*optixDenoiserComputeAverageColor)(OptixDenoiser       handle,
345                                                     CUstream            stream,
346                                                     const OptixImage2D* inputImage,
347                                                     CUdeviceptr         outputAverageColor,
348                                                     CUdeviceptr         scratch,
349                                                     size_t              scratchSizeInBytes);
350
352     OptixResult (*optixDenoiserCreateWithUserModel)(OptixDeviceContext context, const void * data, size_t
dataSizeInBytes, OptixDenoiser* returnHandle);
353     //@ }
354
355 } OptixFunctionTable;
356   // end group optix_function_table
358
359 #ifdef __cplusplus
360 }
361 #endif
362
363 #endif /* OPTIX_DEFINE_ABI_VERSION_ONLY */
364
365 #endif /* __optix_optix_function_table_h__ */
```

8.21 optix_function_table_definition.h File Reference

Variables

- OptixFunctionTable g_optixFunctionTable

8.21.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.22 optix_function_table_definition.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
3 *
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly
8 * prohibited.
9 *
```

```
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
24
25 #ifndef __optix_optix_function_table_definition_h__
26 #define __optix_optix_function_table_definition_h__
27
28 #include "optix_function_table.h"
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
41 OptixFunctionTable g_optixFunctionTable;
42   // end group optix_function_table
44
45 #ifdef __cplusplus
46 }
47 #endif
48
49 #endif  // __optix_optix_function_table_definition_h__
```

8.23 optix_host.h File Reference

Macros

- #define __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_HOST_H__

8.23.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Host side

8.23.2 Macro Definition Documentation

8.23.2.1 __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_HOST_H__

```
#define __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_HOST_H__
```

8.24 optix_host.h

Go to the documentation of this file.

```
1
2 /*
3 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
4 *
5 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
6 * rights in and to this software, related documentation and any modifications thereto.
7 * Any use, reproduction, disclosure or distribution of this software and related
```

```
8 * documentation without an express license agreement from NVIDIA Corporation is strictly
9 * prohibited.
10 *
11 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
12 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
13 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
14 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
15 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
16 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
17 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
18 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
19 * SUCH DAMAGES
20 */
21
30 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
31 #  define __OPTIX_INCLUDE_INTERNAL_HEADERS__
32 #  define __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_HOST_H__
33 #endif
34 #include "optix_7_host.h"
35 #if defined(__UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_HOST_H__)
36 #  undef __OPTIX_INCLUDE_INTERNAL_HEADERS__
37 #  undef __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_HOST_H__
38 #endif
```

8.25 optix_stack_size.h File Reference

Functions

- OptixResult optixUtilAccumulateStackSizes (OptixProgramGroup programGroup, OptixStackSizes *stackSizes)
- 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)

8.25.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.26 optix_stack_size.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
3 *
4 * Redistribution and use in source and binary forms, with or without
5 * modification, are permitted provided that the following conditions
6 * are met:
7 *  * Redistributions of source code must retain the above copyright
8 *    notice, this list of conditions and the following disclaimer.
9 *  * Redistributions in binary form must reproduce the above copyright
10 *    notice, this list of conditions and the following disclaimer in the
11 *    documentation and/or other materials provided with the distribution.
12 *  * Neither the name of NVIDIA CORPORATION nor the names of its
13 *    contributors may be used to endorse or promote products derived
14 *    from this software without specific prior written permission.
15 *
16 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS “AS IS” AND ANY
17 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
18 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
19 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
20 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
21 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
22 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
23 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
24 * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
25 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
26 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
27 */
28
32
33 #ifndef __optix_optix_stack_size_h__
34 #define __optix_optix_stack_size_h__
35
36 #include "optix.h"
37
38 #include <algorithm>
39 #include <cstring>
40
41 #ifdef __cplusplus
42 extern "C" {
43 #endif
44
52 inline OptixResult optixUtilAccumulateStackSizes(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes)
53 {
54     if(!stackSizes)
55         return OPTIX_ERROR_INVALID_VALUE;
56
57     OptixStackSizes localStackSizes;
58     OptixResult     result = optixProgramGroupGetStackSize(programGroup, &localStackSizes);
59     if(result != OPTIX_SUCCESS)
60         return result;
61
62     stackSizes->cssRG = std::max(stackSizes->cssRG, localStackSizes.cssRG);
63     stackSizes->cssMS = std::max(stackSizes->cssMS, localStackSizes.cssMS);
64     stackSizes->cssCH = std::max(stackSizes->cssCH, localStackSizes.cssCH);
65     stackSizes->cssAH = std::max(stackSizes->cssAH, localStackSizes.cssAH);
66     stackSizes->cssIS = std::max(stackSizes->cssIS, localStackSizes.cssIS);
67     stackSizes->cssCC = std::max(stackSizes->cssCC, localStackSizes.cssCC);
68     stackSizes->dssDC = std::max(stackSizes->dssDC, localStackSizes.dssDC);
```

```
69
70      return OPTIX_SUCCESS;
71  }
72
86  inline OptixResult optixUtilComputeStackSizes(const OptixStackSizes* stackSizes,
87                                                 unsigned int           maxTraceDepth,
88                                                 unsigned int           maxCCDepth,
89                                                 unsigned int           maxDCDepth,
90                                                unsigned int*          directCallableStackSizeFromTraversal,
91                                                 unsigned int*          directCallableStackSizeFromState,
92                                                 unsigned int*          continuationStackSize)
93  {
94      if(!stackSizes)
95          return OPTIX_ERROR_INVALID_VALUE;
96
97      const unsigned int cssRG = stackSizes->cssRG;
98      const unsigned int cssMS = stackSizes->cssMS;
99      const unsigned int cssCH = stackSizes->cssCH;
100      const unsigned int cssAH = stackSizes->cssAH;
101      const unsigned int cssIS = stackSizes->cssIS;
102      const unsigned int cssCC = stackSizes->cssCC;
103      const unsigned int dssDC = stackSizes->dssDC;
104
105      if(directCallableStackSizeFromTraversal)
106          *directCallableStackSizeFromTraversal = maxDCDepth * dssDC;
107      if(directCallableStackSizeFromState)
108          *directCallableStackSizeFromState = maxDCDepth * dssDC;
109
110      // upper bound on continuation stack used by call trees of continuation callables
111      unsigned int cssCCTree = maxCCDepth * cssCC;
112
113      // upper bound on continuation stack used by CH or MS programs including the call tree of
114      // continuation callables
115      unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
116
117      // clang-format off
118      if(continuationStackSize)
119          *continuationStackSize
120              = cssRG + cssCCTree
121              + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
122              + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
123      // clang-format on
124
125      return OPTIX_SUCCESS;
126  }
127
151 inline OptixResult optixUtilComputeStackSizesDCSplit(const OptixStackSizes* stackSizes,
152                                                       unsigned int           dssDCFromTraversal,
153                                                       unsigned int           dssDCFromState,
154                                                       unsigned int           maxTraceDepth,
155                                                       unsigned int           maxCCDepth,
156                                                       unsigned int           maxDCDepthFromTraversal,
157                                                       unsigned int           maxDCDepthFromState,
158                                                       unsigned int*
directCallableStackSizeFromTraversal,
159                                                  unsigned int*          directCallableStackSizeFromState,
160                                                       unsigned int*          continuationStackSize)
161 {
162      if(!stackSizes)
163          return OPTIX_ERROR_INVALID_VALUE;
164
165      const unsigned int cssRG = stackSizes->cssRG;
166      const unsigned int cssMS = stackSizes->cssMS;
167      const unsigned int cssCH = stackSizes->cssCH;
168      const unsigned int cssAH = stackSizes->cssAH;
169      const unsigned int cssIS = stackSizes->cssIS;
170      const unsigned int cssCC = stackSizes->cssCC;
```

```
171     // use dssDCFromTraversal and dssDCFromState instead of stackSizes->dssDC
172
173     if(directCallableStackSizeFromTraversal)
174         *directCallableStackSizeFromTraversal = maxDCDepthFromTraversal * dssDCFromTraversal;
175     if(directCallableStackSizeFromState)
176         *directCallableStackSizeFromState = maxDCDepthFromState * dssDCFromState;
177
178     // upper bound on continuation stack used by call trees of continuation callables
179     unsigned int cssCCTree = maxCCDepth * cssCC;
180
181     // upper bound on continuation stack used by CH or MS programs including the call tree of
182     // continuation callables
183     unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
184
185     // clang-format off
186     if(continuationStackSize)
187         *continuationStackSize
188             = cssRG + cssCCTree
189             + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
190             + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
191     // clang-format on
192
193     return OPTIX_SUCCESS;
194 }
195
212 inline OptixResult optixUtilComputeStackSizesCssCCTree(const OptixStackSizes* stackSizes,
213                                                        unsigned int           cssCCTree,
214                                                        unsigned int           maxTraceDepth,
215                                                        unsigned int           maxDCDepth,
216                                                        unsigned int*
directCallableStackSizeFromTraversal,
217                                                    unsigned int*         directCallableStackSizeFromState,
218                                                        unsigned int*          continuationStackSize)
219 {
220     if(!stackSizes)
221         return OPTIX_ERROR_INVALID_VALUE;
222
223     const unsigned int cssRG = stackSizes->cssRG;
224     const unsigned int cssMS = stackSizes->cssMS;
225     const unsigned int cssCH = stackSizes->cssCH;
226     const unsigned int cssAH = stackSizes->cssAH;
227     const unsigned int cssIS = stackSizes->cssIS;
228     // use cssCCTree instead of stackSizes->cssCC and maxCCDepth
229     const unsigned int dssDC = stackSizes->dssDC;
230
231     if(directCallableStackSizeFromTraversal)
232         *directCallableStackSizeFromTraversal = maxDCDepth * dssDC;
233     if(directCallableStackSizeFromState)
234         *directCallableStackSizeFromState = maxDCDepth * dssDC;
235
236     // upper bound on continuation stack used by CH or MS programs including the call tree of
237     // continuation callables
238     unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
239
240     // clang-format off
241     if(continuationStackSize)
242         *continuationStackSize
243             = cssRG + cssCCTree
244             + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
245             + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
246     // clang-format on
247
248     return OPTIX_SUCCESS;
249 }
250
263 inline OptixResult optixUtilComputeStackSizesSimplePathTracer(OptixProgramGroup        programGroupRG,
264                                                               OptixProgramGroup        programGroupMS1,
```

```
265                                                         const OptixProgramGroup* programGroupCH1,
266                                                   unsigned int            programGroupCH1Count,
267                                                         OptixProgramGroup        programGroupMS2,
268                                                         const OptixProgramGroup* programGroupCH2,
269                                                   unsigned int            programGroupCH2Count,
270                                                         unsigned int*
directCallableStackSizeFromTraversal,
271                                                 unsigned int* directCallableStackSizeFromState,
272                                                         unsigned int* continuationStackSize)
273 {
274     if(!programGroupCH1 && (programGroupCH1Count > 0))
275         return OPTIX_ERROR_INVALID_VALUE;
276     if(!programGroupCH2 && (programGroupCH2Count > 0))
277         return OPTIX_ERROR_INVALID_VALUE;
278
279     OptixResult result;
280
281     OptixStackSizes stackSizesRG = {};
282     result                       = optixProgramGroupGetStackSize(programGroupRG, &stackSizesRG);
283     if(result != OPTIX_SUCCESS)
284         return result;
285
286     OptixStackSizes stackSizesMS1 = {};
287     result                        = optixProgramGroupGetStackSize(programGroupMS1, &stackSizesMS1);
288     if(result != OPTIX_SUCCESS)
289         return result;
290
291     OptixStackSizes stackSizesCH1 = {};
292     for(unsigned int i = 0; i < programGroupCH1Count; ++i)
293     {
294         result = optixUtilAccumulateStackSizes(programGroupCH1[i], &stackSizesCH1);
295         if(result != OPTIX_SUCCESS)
296             return result;
297     }
298
299     OptixStackSizes stackSizesMS2 = {};
300     result                        = optixProgramGroupGetStackSize(programGroupMS2, &stackSizesMS2);
301     if(result != OPTIX_SUCCESS)
302         return result;
303
304     OptixStackSizes stackSizesCH2 = {};
305     memset(&stackSizesCH2, 0, sizeof(OptixStackSizes));
306     for(unsigned int i = 0; i < programGroupCH2Count; ++i)
307     {
308         result = optixUtilAccumulateStackSizes(programGroupCH2[i], &stackSizesCH2);
309         if(result != OPTIX_SUCCESS)
310             return result;
311     }
312
313     const unsigned int cssRG  = stackSizesRG.cssRG;
314     const unsigned int cssMS1 = stackSizesMS1.cssMS;
315     const unsigned int cssCH1 = stackSizesCH1.cssCH;
316     const unsigned int cssMS2 = stackSizesMS2.cssMS;
317     const unsigned int cssCH2 = stackSizesCH2.cssCH;
318     // no AH, IS, CC, or DC programs
319
320     if(directCallableStackSizeFromTraversal)
321         *directCallableStackSizeFromTraversal = 0;
322     if(directCallableStackSizeFromState)
323         *directCallableStackSizeFromState = 0;
324
325     if(continuationStackSize)
326         *continuationStackSize = cssRG + std::max(cssMS1, cssCH1 + std::max(cssMS2, cssCH2));
327
328     return OPTIX_SUCCESS;
329 }
330   // end group optix_utilities
```

```
332
333 #ifdef __cplusplus
334 }
335 #endif
336
337 #endif  // __optix_optix_stack_size_h__
```

8.27 optix_stubs.h File Reference

Macros

- #define WIN32_LEAN_AND_MEAN 1

Functions

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

Variables

- OptixFunctionTable g_optixFunctionTable

8.27.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.27.2 Macro Definition Documentation

8.27.2.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN 1
```

8.27.3 Function Documentation

8.27.3.1 optixLoadWindowsDll()

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

8.27.3.2 optixLoadWindowsDllFromName()

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

8.28 optix_stubs.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
3 *
4 * Redistribution and use in source and binary forms, with or without
```

```
5 * modification, are permitted provided that the following conditions
6 * are met:
7 *  * Redistributions of source code must retain the above copyright
8 *    notice, this list of conditions and the following disclaimer.
9 *  * Redistributions in binary form must reproduce the above copyright
10 *    notice, this list of conditions and the following disclaimer in the
11 *    documentation and/or other materials provided with the distribution.
12 *  * Neither the name of NVIDIA CORPORATION nor the names of its
13 *    contributors may be used to endorse or promote products derived
14 *    from this software without specific prior written permission.
15 *
16 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS “AS IS” AND ANY
17 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
18 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
19 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
20 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
21 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
22 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
23 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
24 * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
25 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
26 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
27 */
28
32
33 #ifndef __optix_optix_stubs_h__
34 #define __optix_optix_stubs_h__
35
36 #include "optix_function_table.h"
37
38 #ifdef _WIN32
39 #ifndef WIN32_LEAN_AND_MEAN
40 #define WIN32_LEAN_AND_MEAN 1
41 #endif
42 #include <windows.h>
43 // The cfgmgr32 header is necessary for interrogating driver information in the registry.
44 // For convenience the library is also linked in automatically using the #pragma command.
45 #include <cfgmgr32.h>
46 #pragma comment(lib, "Cfgmgr32.lib")
47 #include <string.h>
48 #else
49 #include <dlfcn.h>
50 #endif
51
52 #ifdef __cplusplus
53 extern "C" {
54 #endif
55
56 // The function table needs to be defined in exactly one translation unit. This can be
57 // achieved by including optix_function_table_definition.h in that translation unit.
58 extern OptixFunctionTable g_optixFunctionTable;
59
60 #ifdef _WIN32
61 #if defined(_MSC_VER)
62 // Visual Studio produces warnings suggesting strcpy and friends being replaced with _s
63 // variants. All the string lengths and allocation sizes have been calculated and should
64 // be safe, so we are disabling this warning to increase compatibility.
65 #    pragma warning(push)
66 #    pragma warning(disable : 4996)
67 #endif
68 static void* optixLoadWindowsDllFromName(const char* optixDllName)
69 {
70     void* handle = NULL;
71
72     // Try the bare dll name first.  This picks it up in the local path, followed by
73     //  standard Windows paths.
74     handle = LoadLibraryA((LPSTR)optixDllName);
```

```
75      if(handle)
76          return handle;
77  // If we don't find it in the default dll search path, try the system paths
78
79      // Get the size of the path first, then allocate
80      unsigned int size = GetSystemDirectoryA(NULL, 0);
81      if(size == 0)
82      {
83          // Couldn't get the system path size, so bail
84          return NULL;
85      }
86      size_t pathSize   = size + 1 + strlen(optixDllName);
87      char*  systemPath = (char*)malloc(pathSize);
88      if(systemPath == NULL)
89          return NULL;
90      if(GetSystemDirectoryA(systemPath, size) != size - 1)
91      {
92          // Something went wrong
93          free(systemPath);
94          return NULL;
95      }
96      strcat(systemPath, "\\");
97      strcat(systemPath, optixDllName);
98      handle = LoadLibraryA(systemPath);
99      free(systemPath);
100      if(handle)
101          return handle;
102
103      // If we didn't find it, go looking in the register store.  Since nvoptix.dll doesn't
104      // have its own registry entry, we are going to look for the opengl driver which lives
105      // next to nvoptix.dll.  0 (null) will be returned if any errors occured.
106
107      static const char* deviceInstanceIdentifiersGUID = "{4d36e968-e325-11ce-bfc1-08002be10318}";
108      const ULONG        flags                         = CM_GETIDLIST_FILTER_CLASS |
CM_GETIDLIST_FILTER_PRESENT;
109      ULONG              deviceListSize                = 0;
110      if(CM_Get_Device_ID_List_SizeA(&deviceListSize, deviceInstanceIdentifiersGUID, flags) != CR_SUCCESS)
111      {
112          return NULL;
113      }
114      char* deviceNames = (char*)malloc(deviceListSize);
115      if(deviceNames == NULL)
116          return NULL;
117      if(CM_Get_Device_ID_ListA(deviceInstanceIdentifiersGUID, deviceNames, deviceListSize, flags))
118      {
119          free(deviceNames);
120          return NULL;
121      }
122      DEVINST devID   = 0;
123      char*   dllPath = NULL;
124
125      // Continue to the next device if errors are encountered.
126      for(char* deviceName = deviceNames; *deviceName; deviceName += strlen(deviceName) + 1)
127      {
128          if(CM_Locate_DevNodeA(&devID, deviceName, CM_LOCATE_DEVNODE_NORMAL) != CR_SUCCESS)
129          {
130              continue;
131          }
132          HKEY regKey = 0;
133          if(CM_Open_DevNode_Key(devID, KEY_QUERY_VALUE, 0, RegDisposition_OpenExisting, &regKey,
CM_REGISTRY_SOFTWARE) != CR_SUCCESS)
134          {
135              continue;
136          }
137          const char* valueName = "OpenGLDriverName";
138          DWORD       valueSize = 0;
139          LSTATUS     ret       = RegQueryValueExA(regKey, valueName, NULL, NULL, NULL, &valueSize);
```

```
140         if(ret != ERROR_SUCCESS)
141         {
142             RegCloseKey(regKey);
143             continue;
144         }
145         char* regValue = (char*)malloc(valueSize);
146         if(regValue == NULL)
147         {
148             RegCloseKey(regKey);
149             continue;
150         }
151         ret            = RegQueryValueExA(regKey, valueName, NULL, NULL, (LPBYTE)regValue, &valueSize);
152         if(ret != ERROR_SUCCESS)
153         {
154             free(regValue);
155             RegCloseKey(regKey);
156             continue;
157         }
158         // Strip the opengl driver dll name from the string then create a new string with
159         // the path and the nvoptix.dll name
160         for(int i = (int) valueSize - 1; i >= 0 && regValue[i] != '\\'; --i)
161             regValue[i] = '\0';
162         size_t newPathSize = strlen(regValue) + strlen(optixDllName) + 1;
163         dllPath            = (char*)malloc(newPathSize);
164         if(dllPath == NULL)
165         {
166             free(regValue);
167             RegCloseKey(regKey);
168             continue;
169         }
170         strcpy(dllPath, regValue);
171         strcat(dllPath, optixDllName);
172         free(regValue);
173         RegCloseKey(regKey);
174         handle = LoadLibraryA((LPCSTR)dllPath);
175         free(dllPath);
176         if(handle)
177             break;
178     }
179     free(deviceNames);
180     return handle;
181 }
182 #if defined(_MSC_VER)
183 #    pragma warning(pop)
184 #endif
185
186 static void* optixLoadWindowsDll()
187 {
188     return optixLoadWindowsDllFromName("nvoptix.dll");
189 }
190 #endif
191
194
204 inline OptixResult optixInitWithHandle(void** handlePtr)
205 {
206     // Make sure these functions get initialized to zero in case the DLL and function
207     // table can't be loaded
208     g_optixFunctionTable.optixGetErrorName   = 0;
209     g_optixFunctionTable.optixGetErrorString = 0;
210
211     if(!handlePtr)
212         return OPTIX_ERROR_INVALID_VALUE;
213
214 #ifdef _WIN32
215     *handlePtr = optixLoadWindowsDll();
216     if(!*handlePtr)
217         return OPTIX_ERROR_LIBRARY_NOT_FOUND;
```

```
218
219     void* symbol = GetProcAddress((HMODULE)*handlePtr, "optixQueryFunctionTable");
220     if(!symbol)
221         return OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND;
222 #else
223     *handlePtr = dlopen("libnvoptix.so.1", RTLD_NOW);
224     if(!*handlePtr)
225         return OPTIX_ERROR_LIBRARY_NOT_FOUND;
226
227     void* symbol = dlsym(*handlePtr, "optixQueryFunctionTable");
228     if(!symbol)
229         return OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND;
230 #endif
231
232     OptixQueryFunctionTable_t* optixQueryFunctionTable = (OptixQueryFunctionTable_t*)symbol;
233
234     return optixQueryFunctionTable(OPTIX_ABI_VERSION, 0, 0, 0, &g_optixFunctionTable,
sizeof(g_optixFunctionTable));
235 }
236
240 inline OptixResult optixInit(void)
241 {
242     void* handle;
243     return optixInitWithHandle(&handle);
244 }
245
251 inline OptixResult optixUninitWithHandle(void* handle)
252 {
253     if(!handle)
254       return OPTIX_ERROR_INVALID_VALUE;
255 #ifdef _WIN32
256     if(!FreeLibrary((HMODULE)handle))
257         return OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE;
258 #else
259     if(dlclose(handle))
260         return OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE;
261 #endif
262     OptixFunctionTable empty = { 0 };
263     g_optixFunctionTable = empty;
264     return OPTIX_SUCCESS;
265 }
266
267   // end group optix_utilities
269
270 #ifndef OPTIX_DOXYGEN_SHOULD_SKIP_THIS
271
272 // Stub functions that forward calls to the corresponding function pointer in the function table.
273
274 inline const char* optixGetErrorName(OptixResult result)
275 {
276     if(g_optixFunctionTable.optixGetErrorName)
277         return g_optixFunctionTable.optixGetErrorName(result);
278
279     // If the DLL and symbol table couldn't be loaded, provide a set of error strings
280     // suitable for processing errors related to the DLL loading.
281     switch(result)
282     {
283         case OPTIX_SUCCESS:
284             return "OPTIX_SUCCESS";
285         case OPTIX_ERROR_INVALID_VALUE:
286             return "OPTIX_ERROR_INVALID_VALUE";
287         case OPTIX_ERROR_UNSUPPORTED_ABI_VERSION:
288             return "OPTIX_ERROR_UNSUPPORTED_ABI_VERSION";
289         case OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH:
290             return "OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH";
291         case OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS:
292             return "OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS";
```

```
293         case OPTIX_ERROR_LIBRARY_NOT_FOUND:
294             return "OPTIX_ERROR_LIBRARY_NOT_FOUND";
295         case OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND:
296             return "OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND";
297         case OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE:
298             return "OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE";
299         default:
300             return "Unknown OptixResult code";
301     }
302 }
303
304 inline const char* optixGetErrorString(OptixResult result)
305 {
306     if(g_optixFunctionTable.optixGetErrorString)
307         return g_optixFunctionTable.optixGetErrorString(result);
308
309     // If the DLL and symbol table couldn't be loaded, provide a set of error strings
310     // suitable for processing errors related to the DLL loading.
311     switch(result)
312     {
313         case OPTIX_SUCCESS:
314             return "Success";
315         case OPTIX_ERROR_INVALID_VALUE:
316             return "Invalid value";
317         case OPTIX_ERROR_UNSUPPORTED_ABI_VERSION:
318             return "Unsupported ABI version";
319         case OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH:
320             return "Function table size mismatch";
321         case OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS:
322             return "Invalid options to entry function";
323         case OPTIX_ERROR_LIBRARY_NOT_FOUND:
324             return "Library not found";
325         case OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND:
326             return "Entry symbol not found";
327         case OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE:
328             return "Library could not be unloaded";
329         default:
330             return "Unknown OptixResult code";
331     }
332 }
333
334 inline OptixResult optixDeviceContextCreate(CUcontext fromContext, const OptixDeviceContextOptions*
options, OptixDeviceContext* context)
335 {
336     return g_optixFunctionTable.optixDeviceContextCreate(fromContext, options, context);
337 }
338
339 inline OptixResult optixDeviceContextDestroy(OptixDeviceContext context)
340 {
341     return g_optixFunctionTable.optixDeviceContextDestroy(context);
342 }
343
344 inline OptixResult optixDeviceContextGetProperty(OptixDeviceContext context, OptixDeviceProperty
property, void* value, size_t sizeInBytes)
345 {
346     return g_optixFunctionTable.optixDeviceContextGetProperty(context, property, value, sizeInBytes);
347 }
348
349 inline OptixResult optixDeviceContextSetLogCallback(OptixDeviceContext context,
350                                                     OptixLogCallback   callbackFunction,
351                                                     void*              callbackData,
352                                                     unsigned int       callbackLevel)
353 {
354     return g_optixFunctionTable.optixDeviceContextSetLogCallback(context, callbackFunction,
callbackData, callbackLevel);
355 }
356
```

```
357 inline OptixResult optixDeviceContextSetCacheEnabled(OptixDeviceContext context, int enabled)
358 {
359     return g_optixFunctionTable.optixDeviceContextSetCacheEnabled(context, enabled);
360 }
361
362 inline OptixResult optixDeviceContextSetCacheLocation(OptixDeviceContext context, const char* location)
363 {
364     return g_optixFunctionTable.optixDeviceContextSetCacheLocation(context, location);
365 }
366
367 inline OptixResult optixDeviceContextSetCacheDatabaseSizes(OptixDeviceContext context, size_t
lowWaterMark, size_t highWaterMark)
368 {
369     return g_optixFunctionTable.optixDeviceContextSetCacheDatabaseSizes(context, lowWaterMark,
highWaterMark);
370 }
371
372 inline OptixResult optixDeviceContextGetCacheEnabled(OptixDeviceContext context, int* enabled)
373 {
374     return g_optixFunctionTable.optixDeviceContextGetCacheEnabled(context, enabled);
375 }
376
377 inline OptixResult optixDeviceContextGetCacheLocation(OptixDeviceContext context, char* location, size_t
locationSize)
378 {
379     return g_optixFunctionTable.optixDeviceContextGetCacheLocation(context, location, locationSize);
380 }
381
382 inline OptixResult optixDeviceContextGetCacheDatabaseSizes(OptixDeviceContext context, size_t*
lowWaterMark, size_t* highWaterMark)
383 {
384     return g_optixFunctionTable.optixDeviceContextGetCacheDatabaseSizes(context, lowWaterMark,
highWaterMark);
385 }
386
387 inline OptixResult optixModuleCreateFromPTX(OptixDeviceContext                 context,
388                                             const OptixModuleCompileOptions*   moduleCompileOptions,
389                                             const OptixPipelineCompileOptions* pipelineCompileOptions,
390                                             const char*                        PTX,
391                                             size_t                             PTXsize,
392                                             char*                              logString,
393                                             size_t*                            logStringSize,
394                                             OptixModule*                       module)
395 {
396     return g_optixFunctionTable.optixModuleCreateFromPTX(context, moduleCompileOptions,
pipelineCompileOptions, PTX,
397                                                          PTXsize, logString, logStringSize, module);
398 }
399
400 inline OptixResult optixModuleCreateFromPTXWithTasks(OptixDeviceContext                 context,
401                                                      const OptixModuleCompileOptions*   moduleCompileOptions,
402                                                           const OptixPipelineCompileOptions*
pipelineCompileOptions,
403                                                           const char*                        PTX,
404                                                           size_t                             PTXsize,
405                                                           char*                              logString,
406                                                           size_t*                            logStringSize,
407                                                           OptixModule*                       module,
408                                                           OptixTask*                         firstTask)
409 {
410     return g_optixFunctionTable.optixModuleCreateFromPTXWithTasks(context, moduleCompileOptions,
pipelineCompileOptions, PTX,
411                                                                   PTXsize, logString, logStringSize,
module, firstTask);
412 }
413
414 inline OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState* state)
```

```
415 {
416     return g_optixFunctionTable.optixModuleGetCompilationState(module, state);
417 }
418
419 inline OptixResult optixModuleDestroy(OptixModule module)
420 {
421     return g_optixFunctionTable.optixModuleDestroy(module);
422 }
423
424 inline OptixResult optixBuiltinISModuleGet(OptixDeviceContext                 context,
425                                            const OptixModuleCompileOptions*   moduleCompileOptions,
426                                            const OptixPipelineCompileOptions* pipelineCompileOptions,
427                                            const OptixBuiltinISOptions*       builtinISOptions,
428                                            OptixModule*                       builtinModule)
429 {
430     return g_optixFunctionTable.optixBuiltinISModuleGet(context, moduleCompileOptions,
pipelineCompileOptions,
431                                                         builtinISOptions, builtinModule);
432 }
433
434 inline OptixResult optixTaskExecute(OptixTask task, OptixTask* additionalTasks, unsigned int
maxNumAdditionalTasks, unsigned int* numAdditionalTasksCreated)
435 {
436     return g_optixFunctionTable.optixTaskExecute(task, additionalTasks, maxNumAdditionalTasks,
numAdditionalTasksCreated);
437 }
438
439 inline OptixResult optixProgramGroupCreate(OptixDeviceContext              context,
440                                            const OptixProgramGroupDesc*    programDescriptions,
441                                            unsigned int                    numProgramGroups,
442                                            const OptixProgramGroupOptions* options,
443                                            char*                           logString,
444                                            size_t*                         logStringSize,
445                                            OptixProgramGroup*              programGroups)
446 {
447     return g_optixFunctionTable.optixProgramGroupCreate(context, programDescriptions, numProgramGroups,
options,
448                                                         logString, logStringSize, programGroups);
449 }
450
451 inline OptixResult optixProgramGroupDestroy(OptixProgramGroup programGroup)
452 {
453     return g_optixFunctionTable.optixProgramGroupDestroy(programGroup);
454 }
455
456 inline OptixResult optixProgramGroupGetStackSize(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes)
457 {
458     return g_optixFunctionTable.optixProgramGroupGetStackSize(programGroup, stackSizes);
459 }
460
461 inline OptixResult optixPipelineCreate(OptixDeviceContext                 context,
462                                        const OptixPipelineCompileOptions* pipelineCompileOptions,
463                                        const OptixPipelineLinkOptions*    pipelineLinkOptions,
464                                        const OptixProgramGroup*           programGroups,
465                                        unsigned int                       numProgramGroups,
466                                        char*                              logString,
467                                        size_t*                            logStringSize,
468                                        OptixPipeline*                     pipeline)
469 {
470     return g_optixFunctionTable.optixPipelineCreate(context, pipelineCompileOptions,
pipelineLinkOptions, programGroups,
471                                                     numProgramGroups, logString, logStringSize, pipeline);
472 }
473
474 inline OptixResult optixPipelineDestroy(OptixPipeline pipeline)
475 {
```

```
476     return g_optixFunctionTable.optixPipelineDestroy(pipeline);
477 }
478
479 inline OptixResult optixPipelineSetStackSize(OptixPipeline pipeline,
480                                              unsigned int  directCallableStackSizeFromTraversal,
481                                              unsigned int  directCallableStackSizeFromState,
482                                              unsigned int  continuationStackSize,
483                                              unsigned int  maxTraversableGraphDepth)
484 {
485     return g_optixFunctionTable.optixPipelineSetStackSize(pipeline,
directCallableStackSizeFromTraversal, directCallableStackSizeFromState,
486                                                           continuationStackSize, maxTraversableGraphDepth);
487 }
488
489 inline OptixResult optixAccelComputeMemoryUsage(OptixDeviceContext            context,
490                                                 const OptixAccelBuildOptions* accelOptions,
491                                                 const OptixBuildInput*        buildInputs,
492                                                 unsigned int                  numBuildInputs,
493                                                 OptixAccelBufferSizes*        bufferSizes)
494 {
495     return g_optixFunctionTable.optixAccelComputeMemoryUsage(context, accelOptions, buildInputs,
numBuildInputs, bufferSizes);
496 }
497
498 inline OptixResult optixAccelBuild(OptixDeviceContext            context,
499                                    CUstream                      stream,
500                                    const OptixAccelBuildOptions* accelOptions,
501                                    const OptixBuildInput*        buildInputs,
502                                    unsigned int                  numBuildInputs,
503                                    CUdeviceptr                   tempBuffer,
504                                    size_t                        tempBufferSizeInBytes,
505                                    CUdeviceptr                   outputBuffer,
506                                    size_t                        outputBufferSizeInBytes,
507                                    OptixTraversableHandle*       outputHandle,
508                                    const OptixAccelEmitDesc*     emittedProperties,
509                                    unsigned int                  numEmittedProperties)
510 {
511     return g_optixFunctionTable.optixAccelBuild(context, stream, accelOptions, buildInputs,
numBuildInputs, tempBuffer,
512                                                 tempBufferSizeInBytes, outputBuffer, outputBufferSizeInBytes,
513                                                      outputHandle, emittedProperties, numEmittedProperties);
514 }
515
516
517 inline OptixResult optixAccelGetRelocationInfo(OptixDeviceContext context, OptixTraversableHandle
handle, OptixRelocationInfo* info)
518 {
519     return g_optixFunctionTable.optixAccelGetRelocationInfo(context, handle, info);
520 }
521
522
523 inline OptixResult optixCheckRelocationCompatibility(OptixDeviceContext context, const
OptixRelocationInfo* info, int* compatible)
524 {
525     return g_optixFunctionTable.optixCheckRelocationCompatibility(context, info, compatible);
526 }
527
528 inline OptixResult optixAccelRelocate(OptixDeviceContext             context,
529                                       CUstream                       stream,
530                                       const OptixRelocationInfo*     info,
531                                       const OptixRelocateInput*      relocateInputs,
532                                       size_t                         numRelocateInputs,
533                                       CUdeviceptr                    targetAccel,
534                                       size_t                         targetAccelSizeInBytes,
535                                       OptixTraversableHandle*        targetHandle)
536 {
537     return g_optixFunctionTable.optixAccelRelocate(context, stream, info, relocateInputs,
```

```
numRelocateInputs,
538                                                 targetAccel, targetAccelSizeInBytes, targetHandle);
539 }
540
541 inline OptixResult optixAccelCompact(OptixDeviceContext      context,
542                                      CUstream                stream,
543                                      OptixTraversableHandle  inputHandle,
544                                      CUdeviceptr             outputBuffer,
545                                      size_t                  outputBufferSizeInBytes,
546                                      OptixTraversableHandle* outputHandle)
547 {
548     return g_optixFunctionTable.optixAccelCompact(context, stream, inputHandle, outputBuffer,
outputBufferSizeInBytes, outputHandle);
549 }
550
551 inline OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext      onDevice,
552                                                           CUdeviceptr             pointer,
553                                                           OptixTraversableType    traversableType,
554                                                           OptixTraversableHandle* traversableHandle)
555 {
556     return g_optixFunctionTable.optixConvertPointerToTraversableHandle(onDevice, pointer,
traversableType, traversableHandle);
557 }
558
559 inline OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext
context,
560                                                                const OptixOpacityMicromapArrayBuildInput*
buildInput,
561                                                                  OptixMicromapBufferSizes*
bufferSizes)
562 {
563     return g_optixFunctionTable.optixOpacityMicromapArrayComputeMemoryUsage(context, buildInput,
bufferSizes);
564 }
565
566 inline OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext                        context,
567                                                   CUstream                                  stream,
568                                                  const OptixOpacityMicromapArrayBuildInput* buildInput,
569                                                   const OptixMicromapBuffers*               buffers)
570 {
571     return g_optixFunctionTable.optixOpacityMicromapArrayBuild(context, stream, buildInput, buffers);
572 }
573
574 inline OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext   context,
575                                                               CUdeviceptr          opacityMicromapArray,
576                                                                OptixRelocationInfo* info)
577 {
578     return g_optixFunctionTable.optixOpacityMicromapArrayGetRelocationInfo(context,
opacityMicromapArray, info);
579 }
580
581 inline OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext         context,
582                                                      CUstream                   stream,
583                                                      const OptixRelocationInfo* info,
584                                                     CUdeviceptr                targetOpacityMicromapArray,
585                                                      size_t
targetOpacityMicromapArraySizeInBytes)
586 {
587      return g_optixFunctionTable.optixOpacityMicromapArrayRelocate(context, stream, info,
targetOpacityMicromapArray, targetOpacityMicromapArraySizeInBytes);
588 }
589
590
591 inline OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer)
592 {
593     return g_optixFunctionTable.optixSbtRecordPackHeader(programGroup, sbtRecordHeaderHostPointer);
```

```
594 }
595
596 inline OptixResult optixLaunch(OptixPipeline                pipeline,
597                                CUstream                     stream,
598                                CUdeviceptr                  pipelineParams,
599                                size_t                       pipelineParamsSize,
600                                const OptixShaderBindingTable* sbt,
601                                unsigned int                 width,
602                                unsigned int                 height,
603                                unsigned int                 depth)
604 {
605     return g_optixFunctionTable.optixLaunch(pipeline, stream, pipelineParams, pipelineParamsSize, sbt,
width, height, depth);
606 }
607
608 inline OptixResult optixDenoiserCreate(OptixDeviceContext context, OptixDenoiserModelKind modelKind,
const OptixDenoiserOptions* options, OptixDenoiser* returnHandle)
609 {
610     return g_optixFunctionTable.optixDenoiserCreate(context, modelKind, options, returnHandle);
611 }
612
613 inline OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context, const void* data, size_t
dataSizeInBytes, OptixDenoiser* returnHandle)
614 {
615     return g_optixFunctionTable.optixDenoiserCreateWithUserModel(context, data, dataSizeInBytes,
returnHandle);
616 }
617
618 inline OptixResult optixDenoiserDestroy(OptixDenoiser handle)
619 {
620     return g_optixFunctionTable.optixDenoiserDestroy(handle);
621 }
622
623 inline OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser handle,
624                                                        unsigned int        maximumInputWidth,
625                                                        unsigned int        maximumInputHeight,
626                                                        OptixDenoiserSizes* returnSizes)
627 {
628     return g_optixFunctionTable.optixDenoiserComputeMemoryResources(handle, maximumInputWidth,
maximumInputHeight, returnSizes);
629 }
630
631 inline OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
632                                       CUstream      stream,
633                                       unsigned int  inputWidth,
634                                       unsigned int  inputHeight,
635                                       CUdeviceptr   denoiserState,
636                                       size_t        denoiserStateSizeInBytes,
637                                       CUdeviceptr   scratch,
638                                       size_t        scratchSizeInBytes)
639 {
640     return g_optixFunctionTable.optixDenoiserSetup(denoiser, stream, inputWidth, inputHeight,
denoiserState,
641                                                    denoiserStateSizeInBytes, scratch, scratchSizeInBytes);
642 }
643
644 inline OptixResult optixDenoiserInvoke(OptixDenoiser                   handle,
645                                        CUstream                        stream,
646                                        const OptixDenoiserParams*      params,
647                                        CUdeviceptr                     denoiserData,
648                                        size_t                          denoiserDataSize,
649                                        const OptixDenoiserGuideLayer*  guideLayer,
650                                        const OptixDenoiserLayer*       layers,
651                                        unsigned int                    numLayers,
652                                        unsigned int                    inputOffsetX,
653                                        unsigned int                    inputOffsetY,
654                                        CUdeviceptr                     scratch,
```

```
655                                         size_t                          scratchSizeInBytes)
656 {
657     return g_optixFunctionTable.optixDenoiserInvoke(handle, stream, params, denoiserData,
denoiserDataSize,
658                                                     guideLayer, layers, numLayers,
659                                                 inputOffsetX, inputOffsetY, scratch, scratchSizeInBytes);
660 }
661
662 inline OptixResult optixDenoiserComputeIntensity(OptixDenoiser       handle,
663                                                  CUstream            stream,
664                                                  const OptixImage2D* inputImage,
665                                                  CUdeviceptr         outputIntensity,
666                                                  CUdeviceptr         scratch,
667                                                  size_t              scratchSizeInBytes)
668 {
669     return g_optixFunctionTable.optixDenoiserComputeIntensity(handle, stream, inputImage,
outputIntensity, scratch, scratchSizeInBytes);
670 }
671
672 inline OptixResult optixDenoiserComputeAverageColor(OptixDenoiser       handle,
673                                                     CUstream            stream,
674                                                     const OptixImage2D* inputImage,
675                                                     CUdeviceptr         outputAverageColor,
676                                                     CUdeviceptr         scratch,
677                                                     size_t              scratchSizeInBytes)
678 {
679     return g_optixFunctionTable.optixDenoiserComputeAverageColor(handle, stream, inputImage,
outputAverageColor, scratch, scratchSizeInBytes);
680 }
681
682 #endif  // OPTIX_DOXYGEN_SHOULD_SKIP_THIS
683
684 #ifdef __cplusplus
685 }
686 #endif
687
688 #endif  // __optix_optix_stubs_h__
```

8.29 optix_types.h File Reference

Macros

- #define __OPTIX_INCLUDE_INTERNAL_HEADERS__
- #define __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_TYPES_H__

8.29.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.29.2 Macro Definition Documentation

8.29.2.1 __OPTIX_INCLUDE_INTERNAL_HEADERS__

```
#define __OPTIX_INCLUDE_INTERNAL_HEADERS__
```

8.29.2.2 __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_TYPES_H__

```
#define __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_TYPES_H__
```

8.30 optix_types.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2021 NVIDIA Corporation.  All rights reserved.
3 *
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly
8 * prohibited.
9 *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
28 #ifndef __optix_optix_types_h__
29 #define __optix_optix_types_h__
30
31 // clang-format off
32 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
33 #  define __OPTIX_INCLUDE_INTERNAL_HEADERS__
34 #  define __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_TYPES_H__
35 #endif
36 #include "optix_7_types.h"
37 #if defined(__UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_TYPES_H__)
38 #  undef __OPTIX_INCLUDE_INTERNAL_HEADERS__
39 #  undef __UNDEF_OPTIX_INCLUDE_INTERNAL_HEADERS_OPTIX_TYPES_H__
40 #endif
41 // clang-format on
42
43 #endif // #ifndef __optix_optix_types_h__
```

8.31 main.dox File Reference