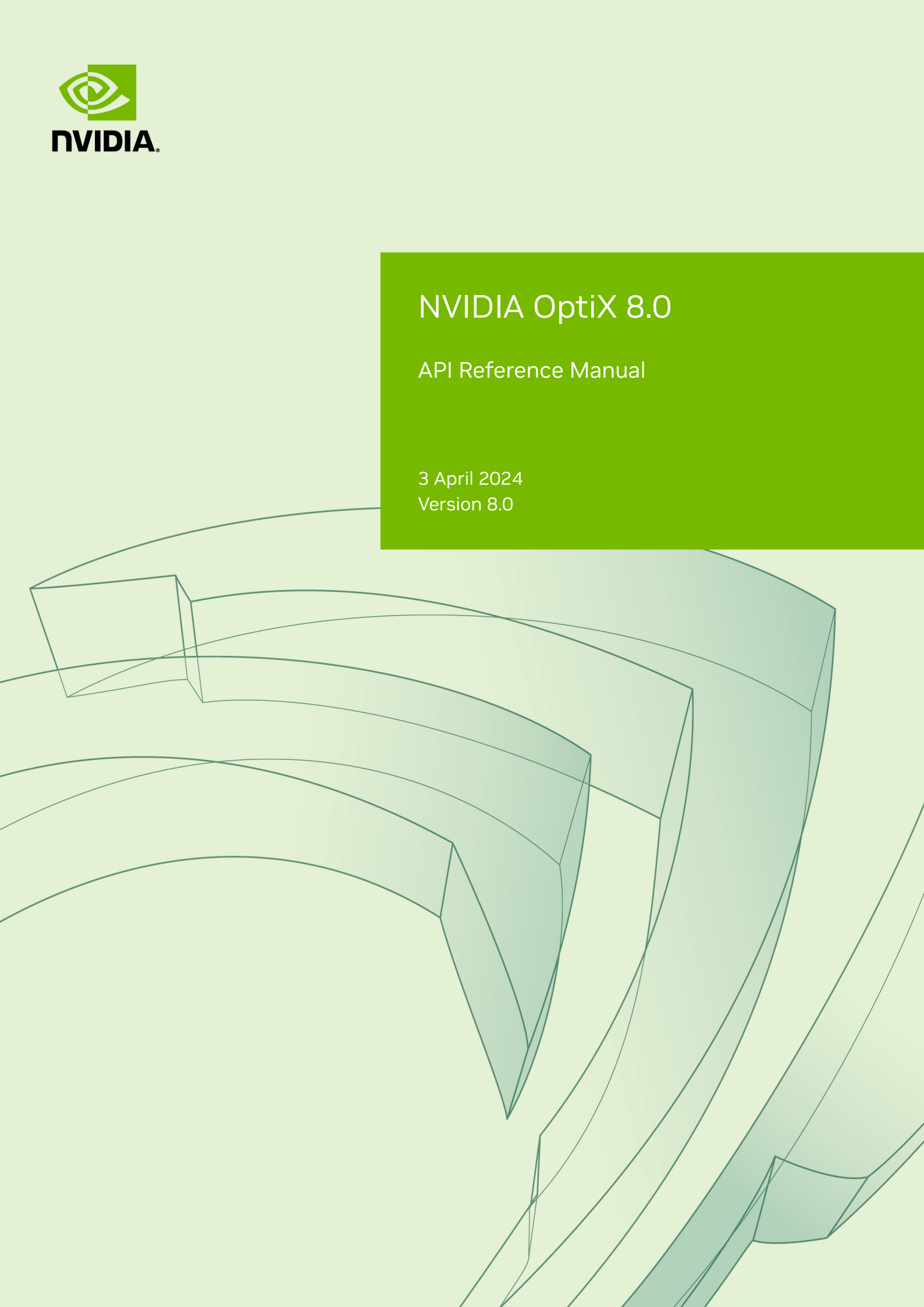

NVIDIA OptiX 8.0

API Reference Manual

3 April 2024
Version 8.0

Table of Contents

1 NVIDIA OptiX 8.0 API

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

2 Module Index

2.1 Modules

Here is a list of all modules:

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

static __forceinline__ __device__ void optixInvoke (OptixPayloadTypeID type, Payload &... payload)

- template<typename... RegAttributes>
  static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int SBToffset, unsigned int SBTstride, unsigned int instIdx, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)
- template<typename... RegAttributes>
  static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int SBToffset, unsigned int SBTstride, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)
- template<typename... RegAttributes>
  static __forceinline__ __device__ void optixMakeHitObjectWithRecord (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int sbtRecordIndex, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)
- static __forceinline__ __device__ void optixMakeMissHitObject (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime)
- static __forceinline__ __device__ void optixMakeNopHitObject ()
- static __forceinline__ __device__ bool optixHitObjectIsHit ()
- static __forceinline__ __device__ bool optixHitObjectIsMiss ()
- static __forceinline__ __device__ bool optixHitObjectIsNop ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex ()
- static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_5 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_6 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_7 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_8 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_9 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_10 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_11 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_12 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_13 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_14 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_15 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_16 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_17 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_18 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_19 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_20 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_21 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_22 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_23 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_24 (unsigned int p)

- static __forceinline__ __device__ void optixSetPayload_25 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_26 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_27 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_28 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_29 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_30 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_31 (unsigned int p)
- static __forceinline__ __device__ unsigned int optixGetPayload_0 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_1 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_2 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_3 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_4 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_5 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_6 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_7 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_8 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_9 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_10 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_11 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_12 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_13 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_14 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_15 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_16 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_17 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_18 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_19 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_20 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_21 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_22 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_23 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_24 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_25 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_26 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_27 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_28 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_29 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_30 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_31 ()
- static __forceinline__ __device__ void optixSetPayloadTypes (unsigned int typeMask)
- static __forceinline__ __device__ unsigned int optixUndefinedValue ()
- static __forceinline__ __device__ float3 optixGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixGetObjectRayOrigin ()
- static __forceinline__ __device__ float3 optixGetObjectRayDirection ()
- static __forceinline__ __device__ float optixGetRayTmin ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmin ()
- static __forceinline__ __device__ float optixGetRayTmax ()

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

- static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTransformType optixGetTransformTypeFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixStaticTransform * optixGetStaticTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixSRTMotionTransform * optixGetSRTMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixMatrixMotionTransform * optixGetMatrixMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceChildFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceInverseTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6, unsigned int a7)
- static __forceinline__ __device__ unsigned int optixGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_7 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2 ()

- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7 ()
- static __forceinline__ __device__ void optixTerminateRay ()
- static __forceinline__ __device__ void optixIgnoreIntersection ()
- static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixGetHitKind ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind ()
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsFrontFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsBackFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ()
- static __forceinline__ __device__ bool optixIsFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsBackFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleHit ()
- static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ()
- static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit ()
- static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleBackFaceHit ()
- static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ()
- static __forceinline__ __device__ float optixGetCurveParameter ()
- static __forceinline__ __device__ float2 optixGetRibbonParameters ()
- static __forceinline__ __device__ uint3 optixGetLaunchIndex ()
- static __forceinline__ __device__ uint3 optixGetLaunchDimensions ()
- static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ()
- static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer ()
- static __forceinline__ __device__ void optixThrowException (int exceptionCode)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)

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

5.1.1 Detailed Description

OptiX Device API.

5.1.2 Function Documentation

5.1.2.1 optixContinuationCall()

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

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

This will call the program that was specified in the OptixProgramGroupCallables ::entryFunctionNameCC in the module specified by OptixProgramGroupCallables::moduleCC.

The address of the SBT entry is calculated by: OptixShaderBindingTable::callablesRecordBase + (OptixShaderBindingTable::callablesRecordStrideInBytes ∗ sbtIndex).

As opposed to direct callable programs, continuation callable programs are allowed to make secondary optixTrace calls.

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

Behavior is undefined if the number of arguments that are being passed in does not match the number of parameters expected by the program that is called. In validation mode an exception will be generated.

Parameters

in	*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.

Available in RG, CH, MS, CC

5.1.2.2 optixDirectCall()

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

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

This will call the program that was specified in the OptixProgramGroupCallables ::entryFunctionNameDC in the module specified by OptixProgramGroupCallables::moduleDC.

The address of the SBT entry is calculated by: OptixShaderBindingTable::callablesRecordBase + (OptixShaderBindingTable::callablesRecordStrideInBytes ∗ sbtIndex).

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

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

Behavior is undefined if the number of arguments that are being passed in does not match the number of parameters expected by the program that is called. In validation mode an exception will be generated.

Parameters

in	*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.

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

5.1.2.3 optixGetAttribute_0()

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

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

Available in AH, CH

5.1.2.4 optixGetAttribute_1()

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

5.1.2.5 optixGetAttribute_2()

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

5.1.2.6 optixGetAttribute_3()

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

5.1.2.7 optixGetAttribute_4()

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

5.1.2.8 optixGetAttribute_5()

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

5.1.2.9 optixGetAttribute_6()

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

5.1.2.10 optixGetAttribute_7()

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

5.1.2.11 optixGetCatmullRomVertexData()

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

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

To access vertex data, the GAS must be built using the flag OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS.

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

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

Available in all OptiX program types

5.1.2.12 optixGetCubicBezierVertexData()

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

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

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

To access vertex data, the GAS must be built using the flag OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS.

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

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

Available in all OptiX program types

5.1.2.13 optixGetCubicBSplineVertexData()

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

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

To access vertex data, the GAS must be built using the flag OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS.

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

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

Available in all OptiX program types

5.1.2.14 optixGetCurveParameter()

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

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

Available in AH, CH

5.1.2.15 optixGetExceptionCode()

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

Returns the exception code.

Available in EX

5.1.2.16 optixGetExceptionDetail_0()

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

[static]

Returns the 32-bit exception detail at slot 0.

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

Available in EX

5.1.2.17 optixGetExceptionDetail_1( )

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

Returns the 32-bit exception detail at slot 1.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.18 optixGetExceptionDetail_2( )

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

Returns the 32-bit exception detail at slot 2.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.19 optixGetExceptionDetail_3( )

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

Returns the 32-bit exception detail at slot 3.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.20 optixGetExceptionDetail_4( )

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

Returns the 32-bit exception detail at slot 4.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.21 optixGetExceptionDetail_5( )

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

Returns the 32-bit exception detail at slot 5.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.22 optixGetExceptionDetail_6( )

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

Returns the 32-bit exception detail at slot 6.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.23 optixGetExceptionDetail_7()

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

Returns the 32-bit exception detail at slot 7.

See also optixGetExceptionDetail_0() Available in EX

5.1.2.24 optixGetExceptionLineInfo()

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

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

The source location is only available for 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.

Available in EX

5.1.2.25 optixGetGASMotionStepCount()

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

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

Available in all OptiX program types

5.1.2.26 optixGetGASMotionTimeBegin()

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

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

Available in all OptiX program types

5.1.2.27 optixGetGASMotionTimeEnd()

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

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

Available in all OptiX program types

5.1.2.28 optixGetGASTraversableHandle()

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

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

Available in IS, AH, CH

5.1.2.29 optixGetHitKind()

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

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

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

Available in AH and CH

5.1.2.30 optixGetInstanceChildFromHandle()

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

Returns child traversable handle from an OptixInstance traversable.

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

Available in all OptiX program types

5.1.2.31 optixGetInstanceId()

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

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

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

Available in IS, AH, CH

5.1.2.32 optixGetInstanceIdFromHandle()

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

Returns instanceId from an OptixInstance traversable.

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

Available in all OptiX program types

5.1.2.33 optixGetInstanceIndex()

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

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

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

Available in IS, AH, CH

5.1.2.34 optixGetInstanceInverseTransformFromHandle()

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

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

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

Available in all OptiX program types

5.1.2.35 optixGetInstanceTransformFromHandle()

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

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

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

Available in all OptiX program types

5.1.2.36 optixGetInstanceTraversableFromIAS()

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

Return the traversable handle of a given instance in an Instance Acceleration Structure (IAS)

To obtain instance traversables by index, the IAS must be built using the flag OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS.

Available in all OptiX program types

5.1.2.37 optixGetLaunchDimensions()

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

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

Available in all OptiX program types

5.1.2.38 optixGetLaunchIndex()

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

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

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

Available in all OptiX program types

5.1.2.39 optixGetLinearCurveVertexData()

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

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

To access vertex data, the GAS must be built using the flag OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS.

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

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

Available in all OptiX program types

5.1.2.40 optixGetMatrixMotionTransformFromHandle()

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

Returns a pointer to a OptixMatrixMotionTransform from its traversable handle.

Returns 0 if the traversable is not of type OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM.

Available in all OptiX program types

5.1.2.41 optixGetMicroTriangleBarycentricsData()

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

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

Available in all OptiX program types

5.1.2.42 optixGetMicroTriangleVertexData()

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

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

Available in all OptiX program types

5.1.2.43 optixGetObjectRayDirection()

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

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

Available in IS and AH

5.1.2.44 optixGetObjectRayOrigin()

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

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

Available in IS and AH

5.1.2.45 optixGetObjectToWorldTransformMatrix()

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

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

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

Available in IS, AH, CH

5.1.2.46 optixGetPayload_0()

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

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

Available in IS, AH, CH, MS

5.1.2.47 optixGetPayload_1()

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

5.1.2.48 optixGetPayload_10()

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

5.1.2.49 optixGetPayload_11()

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

5.1.2.50 optixGetPayload_12()

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

5.1.2.51 optixGetPayload_13()

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

5.1.2.52 optixGetPayload_14()

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

5.1.2.53 optixGetPayload_15()

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

5.1.2.54 optixGetPayload_16()

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

5.1.2.55 optixGetPayload_17()

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

5.1.2.56 optixGetPayload_18()

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

5.1.2.57 optixGetPayload_19()

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

5.1.2.58 optixGetPayload_2()

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

5.1.2.59 optixGetPayload_20()

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

5.1.2.60 optixGetPayload_21()

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

5.1.2.61 optixGetPayload_22()

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

5.1.2.62 optixGetPayload_23()

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

5.1.2.63 optixGetPayload_24()

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

5.1.2.64 optixGetPayload_25()

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

5.1.2.65 optixGetPayload_26()

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

5.1.2.66 optixGetPayload_27()

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

5.1.2.67 optixGetPayload_28()

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

5.1.2.68 optixGetPayload_29()

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

5.1.2.69 optixGetPayload_3()

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

5.1.2.70 optixGetPayload_30()

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

5.1.2.71 optixGetPayload_31()

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

5.1.2.72 optixGetPayload_4()

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

5.1.2.73 optixGetPayload_5()

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

5.1.2.74 optixGetPayload_6()

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

5.1.2.75 optixGetPayload_7()

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

5.1.2.76 optixGetPayload_8()

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

5.1.2.77 optixGetPayload_9()

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

5.1.2.78 optixGetPrimitiveIndex()

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

For a given OptixBuildInputTriangleArray the number of primitives is defined as.

"(OptixBuildInputTriangleArray::indexBuffer == 0) ? OptixBuildInputTriangleArray::numVertices/3 : OptixBuildInputTriangleArray::numIndexTriplets;".

For a given OptixBuildInputCustomPrimitiveArray the number of primitives is defined as numAabbs.

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

In IS and AH this corresponds to the currently intersected primitive.

In CH this corresponds to the primitive index of the closest intersected primitive.

Available in IS, AH, CH, EX

5.1.2.79 optixGetPrimitiveType() [1/2]

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

Function interpreting the hit kind associated with the current optixReportIntersection.

Available in AH, CH

5.1.2.80 optixGetPrimitiveType() [2/2]

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

Function interpreting the result of optixGetHitKind().

Available in all OptiX program types

5.1.2.81 optixGetQuadraticBSplineVertexData()

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

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

To access vertex data, the GAS must be built using the flag OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS.

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

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

Available in all OptiX program types

5.1.2.82 optixGetRayFlags()

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

Returns the rayFlags passed into optixTrace.

Available in IS, AH, CH, MS

5.1.2.83 optixGetRayTime()

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

Returns the rayTime passed into optixTrace.

Returns 0 if motion is disabled.

Available in IS, AH, CH, MS

5.1.2.84 optixGetRayTmax()

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

In IS and CH returns the current smallest reported hitT or the tmax passed into optixTrace if no hit has been reported.

In AH returns the hitT value as passed in to optixReportIntersection

In MS returns the tmax passed into optixTrace

Available in IS, AH, CH, MS

5.1.2.85 optixGetRayTmin()

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

Returns the tmin passed into optixTrace.

Available in IS, AH, CH, MS

5.1.2.86 optixGetRayVisibilityMask()

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

Returns the visibilityMask passed into optixTrace.

Available in IS, AH, CH, MS

5.1.2.87 optixGetRibbonNormal()

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

Return ribbon normal at intersection reported by optixReportIntersection.

Available in all OptiX program types

5.1.2.88 optixGetRibbonParameters()

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

Returns the ribbon parameters along directrix (length) and generator (width) of the current intersection when using OptixBuildInputCurveArray objects with curveType OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE.

Available in AH, CH

5.1.2.89 optixGetRibbonVertexData()

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

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

To access vertex data, the GAS must be built using the flag OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS.

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

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

Available in all OptiX program types

5.1.2.90 optixGetSbtDataPointer()

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

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

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

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

5.1.2.91 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.

Available in IS, AH, CH, EX

5.1.2.92 optixGetSphereData()

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

Return the object space sphere data, center point and radius, in a Geometry Acceleration Structure

(GAS) at a given motion time.

To access sphere data, the GAS must be built using the flag OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS.

data[0] = {x,y,z,w} with {x,y,z} the position of the sphere center and w the radius.

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

Available in all OptiX program types

5.1.2.93 optixGetSRTMotionTransformFromHandle()

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

Returns a pointer to a OptixSRTMotionTransform from its traversable handle.

Returns 0 if the traversable is not of type OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM.

Available in all OptiX program types

5.1.2.94 optixGetStaticTransformFromHandle()

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

Returns a pointer to a OptixStaticTransform from its traversable handle.

Returns 0 if the traversable is not of type OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM.

Available in all OptiX program types

5.1.2.95 optixGetTransformListHandle()

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

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

Available in IS, AH, CH, EX

5.1.2.96 optixGetTransformListSize()

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

Returns the number of transforms on the current transform list.

Available in IS, AH, CH, EX

5.1.2.97 optixGetTransformTypeFromHandle()

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

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

Available in all OptiX program types

5.1.2.98 optixGetTriangleBarycentrics()

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

Convenience function that returns the first two attributes as floats.

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

Available in AH, CH

5.1.2.99 optixGetTriangleVertexData()

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

Return the object space triangle vertex positions of a given triangle in a Geometry Acceleration Structure (GAS) at a given motion time.

To access vertex data, the GAS must be built using the flag OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS.

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

Available in all OptiX program types

5.1.2.100 optixGetWorldRayDirection()

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

Returns the rayDirection passed into optixTrace.

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

Available in IS, AH, CH, MS

5.1.2.101 optixGetWorldRayOrigin()

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

Returns the rayOrigin passed into optixTrace.

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

Available in IS, AH, CH, MS

5.1.2.102 optixGetWorldToObjectTransformMatrix()

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

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

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

Available in IS, AH, CH

5.1.2.103 optixHitObjectGetAttribute_0()

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

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

Results are undefined if the hit object is a miss.

Available in RG, CH, MS, CC, DC

5.1.2.104 optixHitObjectGetAttribute_1()

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

5.1.2.105 optixHitObjectGetAttribute_2()

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

5.1.2.106 optixHitObjectGetAttribute_3()

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

5.1.2.107 optixHitObjectGetAttribute_4()

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

5.1.2.108 optixHitObjectGetAttribute_5()

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

5.1.2.109 optixHitObjectGetAttribute_6()

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

5.1.2.110 optixHitObjectGetAttribute_7()

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

5.1.2.111 optixHitObjectGetHitKind()

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

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

Results are undefined if the hit object is a miss.

See optixGetHitKind( ).

Available in RG, CH, MS, CC, DC

5.1.2.112 optixHitObjectGetInstanceId()

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

Returns the OptixInstance::instanceId of the instance within the top level acceleration structure associated with the outgoing hit object.

Results are undefined if the hit object is a miss.

See optixGetInstanceId( ).

Available in RG, CH, MS, CC, DC

5.1.2.113 optixHitObjectGetInstanceIndex()

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

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

Results are undefined if the hit object is a miss.

See optixGetInstanceIndex( ).

Available in RG, CH, MS, CC, DC

5.1.2.114 optixHitObjectGetPrimitiveIndex()

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

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

Results are undefined if the hit object is a miss.

See optixGetPrimitiveIndex( ) for more details.

Available in RG, CH, MS, CC, DC

5.1.2.115 optixHitObjectGetRayTime()

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.116 optixHitObjectGetRayTmax()

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

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

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

Returns 0 for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.117 optixHitObjectGetRayTmin()

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

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

Returns 0.0f for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.118 optixHitObjectGetSbtDataPointer()

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

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

Returns 0 for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.119 optixHitObjectGetSbtGASIndex()

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

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

Results are undefined if the hit object is a miss.

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

Available in RG, CH, MS, CC, DC

5.1.2.120 optixHitObjectGetSbtRecordIndex()

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

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

Returns 0 for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.121 optixHitObjectGetTransformListHandle()

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

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

Results are undefined if the hit object is a miss.

See optixGetTransformListHandle()

Available in RG, CH, MS, CC, DC

5.1.2.122 optixHitObjectGetTransformListSize()

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

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

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

See optixGetTransformListSize()

Available in RG, CH, MS, CC, DC

5.1.2.123 optixHitObjectGetWorldRayDirection()

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.124 optixHitObjectGetWorldRayOrigin()

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.125 optixHitObjectIsHit()

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

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

Available in RG, CH, MS, CC, DC

5.1.2.126 optixHitObjectIsMiss()

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

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

Available in RG, CH, MS, CC, DC

5.1.2.127 optixHitObjectIsNop()

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

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

Available in RG, CH, MS, CC, DC

5.1.2.128 optixIgnoreIntersection()

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

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

Available in AH

5.1.2.129 optixInvoke() [1/2]

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

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

Parameters

in	*type*	
in,out	*payload*	up to 32 unsigned int values that hold the payload

Available in RG, CH, MS, CC

5.1.2.130 optixInvoke() [2/2]

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

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

Parameters

in,out	*payload*	up to 32 unsigned int values that hold the payload

Available in RG, CH, MS, CC

5.1.2.131 optixIsBackFaceHit() [1/2]

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

Function interpreting the hit kind associated with the current optixReportIntersection.

Available in AH, CH

5.1.2.132 optixIsBackFaceHit() [2/2]

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

Function interpreting the result of optixGetHitKind().

Available in all OptiX program types

5.1.2.133 optixIsDisplacedMicromeshTriangleBackFaceHit()

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

Convenience function interpreting the result of optixGetHitKind().

Available in AH, CH

5.1.2.134 optixIsDisplacedMicromeshTriangleFrontFaceHit()

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

Convenience function interpreting the result of optixGetHitKind().

Available in AH, CH

5.1.2.135 optixIsDisplacedMicromeshTriangleHit()

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

Convenience function interpreting the result of optixGetHitKind().

Available in AH, CH

5.1.2.136 optixIsFrontFaceHit() [1/2]

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

Function interpreting the hit kind associated with the current optixReportIntersection.

Available in AH, CH

5.1.2.137 optixIsFrontFaceHit() [2/2]

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

Function interpreting the result of optixGetHitKind().

Available in all OptiX program types

5.1.2.138 optixIsTriangleBackFaceHit()

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

Convenience function interpreting the result of optixGetHitKind().

Available in AH, CH

5.1.2.139 optixIsTriangleFrontFaceHit()

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

Convenience function interpreting the result of optixGetHitKind().

Available in AH, CH

5.1.2.140 optixIsTriangleHit()

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

Convenience function interpreting the result of optixGetHitKind().

Available in AH, CH

5.1.2.141 optixMakeHitObject() [1/2]

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

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

Parameters

in	*handle*	
in	*rayOrigin*	

Parameters

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

Available in RG, CH, MS, CC

5.1.2.142 optixMakeHitObject() [2/2]

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

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

Parameters

in	*handle*	
in	*rayOrigin*	
in	*rayDirection*	
in	*tmin*	

Parameters

in	*tmax*	
in	*rayTime*	
in	*SBToffset*	really only 4 bits
in	*SBTstride*	really only 4 bits
in	*instIdx*	
in	*sbtGASIdx*	
in	*primIdx*	
in	*hitKind*	
in	*regAttributes*	up to 8 attribute registers

Available in RG, CH, MS, CC

5.1.2.143 optixMakeHitObjectWithRecord()

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

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

Parameters

in	*handle*	
in	*rayOrigin*	
in	*rayDirection*	
in	*tmin*	
in	*tmax*	
in	*rayTime*	

Parameters

in	*sbtRecordIndex*	32 bits
in	*instIdx*	
in	*transforms*	
in	*numTransforms*	
in	*sbtGASIdx*	
in	*primIdx*	
in	*hitKind*	
in	*regAttributes*	up to 8 attribute registers

Available in RG, CH, MS, CC

5.1.2.144 optixMakeMissHitObject()

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

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

Parameters

in	*missSBTIndex*	specifies the miss program invoked on a miss
in	*rayOrigin*	
in	*rayDirection*	
in	*tmin*	
in	*tmax*	
in	*rayTime*	

Available in RG, CH, MS, CC

5.1.2.145 optixMakeNopHitObject()

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

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

Available in RG, CH, MS, CC

5.1.2.146 optixReorder() [1/2]

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

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

Available in RG

5.1.2.147 optixReorder() [2/2]

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

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

Parameters

in	*coherenceHint*
in	*numCoherenceHintBitsFromLSB*

Available in RG

5.1.2.148 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*

Available in IS

5.1.2.149 optixReportIntersection() [2/9]

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

Reports an intersection (overload with 1 attribute register).

See also optixReportIntersection(float,unsigned int) Available in IS

5.1.2.150 optixReportIntersection() [3/9]

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

Reports an intersection (overload with 2 attribute registers).

See also optixReportIntersection(float,unsigned int) Available in IS

5.1.2.151 optixReportIntersection() [4/9]

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

Reports an intersection (overload with 3 attribute registers).

See also optixReportIntersection(float,unsigned int) Available in IS

5.1.2.152 optixReportIntersection() [5/9]

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

Reports an intersection (overload with 4 attribute registers).

See also optixReportIntersection(float,unsigned int) Available in IS

5.1.2.153 optixReportIntersection() [6/9]

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

Reports an intersection (overload with 5 attribute registers).

See also optixReportIntersection(float,unsigned int) Available in IS

5.1.2.154 optixReportIntersection() [7/9]

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

Reports an intersection (overload with 6 attribute registers).

See also optixReportIntersection(float,unsigned int) Available in IS

5.1.2.155 optixReportIntersection() [8/9]

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

Reports an intersection (overload with 7 attribute registers).

See also optixReportIntersection(float,unsigned int) Available in IS

5.1.2.156 optixReportIntersection() [9/9]

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

```
            float hitT,
            unsigned int hitKind,
            unsigned int a0,
            unsigned int a1,
            unsigned int a2,
            unsigned int a3,
            unsigned int a4,
            unsigned int a5,
            unsigned int a6,
            unsigned int a7 )  [static]
```

Reports an intersection (overload with 8 attribute registers).

See also optixReportIntersection(float,unsigned int) Available in IS

5.1.2.157 optixSetPayload_0()

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

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

Available in IS, AH, CH, MS

5.1.2.158 optixSetPayload_1()

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

5.1.2.159 optixSetPayload_10()

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

5.1.2.160 optixSetPayload_11()

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

5.1.2.161 optixSetPayload_12()

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

5.1.2.162 optixSetPayload_13()

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

5.1.2.163 optixSetPayload_14()

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

5.1.2.164 optixSetPayload_15()

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

5.1.2.165 optixSetPayload_16()

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

5.1.2.166 optixSetPayload_17()

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

5.1.2.167 optixSetPayload_18()

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

5.1.2.168 optixSetPayload_19()

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

5.1.2.169 optixSetPayload_2()

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

5.1.2.170 optixSetPayload_20()

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

5.1.2.171 optixSetPayload_21()

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

5.1.2.172 optixSetPayload_22()

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

5.1.2.173 optixSetPayload_23()

```
static __forceinline__ __device__ void optixSetPayload_23 (
```

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

5.1.2.174 optixSetPayload_24()

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

5.1.2.175 optixSetPayload_25()

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

5.1.2.176 optixSetPayload_26()

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

5.1.2.177 optixSetPayload_27()

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

5.1.2.178 optixSetPayload_28()

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

5.1.2.179 optixSetPayload_29()

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

5.1.2.180 optixSetPayload_3()

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

5.1.2.181 optixSetPayload_30()

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

5.1.2.182 optixSetPayload_31()

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

5.1.2.183 optixSetPayload_4()

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

5.1.2.184 optixSetPayload_5()

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

5.1.2.185 optixSetPayload_6()

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

5.1.2.186 optixSetPayload_7()

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

5.1.2.187 optixSetPayload_8()

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

5.1.2.188 optixSetPayload_9()

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

5.1.2.189 optixSetPayloadTypes()

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

Specify the supported payload types for a program.

The supported types are specified as a bitwise combination of payload types. (See OptixPayloadTypeID) May only be called once per program.

Must be called at the top of the program.

Available in IS, AH, CH, MS

5.1.2.190 optixTerminateRay()

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

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

Available in AH

5.1.2.191 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.

Available anywhere

5.1.2.192 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*) Available anywhere

5.1.2.193 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)

Parameters

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∗) Available anywhere

5.1.2.194 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.

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

5.1.2.195 optixThrowException() [2/9]

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

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

See also optixThrowException(int) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.196 optixThrowException() [3/9]

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

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

See also optixThrowException(int) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.197 optixThrowException() [4/9]

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

```
          unsigned int exceptionDetail1,
          unsigned int exceptionDetail2 )  [static]
```

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

See also optixThrowException(int) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.198 optixThrowException() [5/9]

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

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

See also optixThrowException(int) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.199 optixThrowException() [6/9]

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

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

See also optixThrowException(int) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.200 optixThrowException() [7/9]

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

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

See also optixThrowException(int) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.201 optixThrowException() [8/9]

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

```
            unsigned int exceptionDetail0,
            unsigned int exceptionDetail1,
            unsigned int exceptionDetail2,
            unsigned int exceptionDetail3,
            unsigned int exceptionDetail4,
            unsigned int exceptionDetail5,
            unsigned int exceptionDetail6 )  [static]
```

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

See also optixThrowException(int) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.202 optixThrowException() [9/9]

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

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

See also optixThrowException(int) Available in RG, IS, AH, CH, MS, DC, CC

5.1.2.203 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 16 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

Available in RG, CH, MS, CC, DC

5.1.2.204 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*	

Parameters

in	*rayTime*	
in	*visibilityMask*	really only 8 bits
in	*rayFlags*	really only 16 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

Available in RG, CH, MS, CC, DC

5.1.2.205 optixTransformNormalFromObjectToWorldSpace()

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

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

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

Available in IS, AH, CH

5.1.2.206 optixTransformNormalFromWorldToObjectSpace()

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

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

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

Available in IS, AH, CH

5.1.2.207 optixTransformPointFromObjectToWorldSpace()

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

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

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

Available in IS, AH, CH

5.1.2.208 optixTransformPointFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixTransformPointFromWorldToObjectSpace (
```

```
          float3 point ) [static]
```

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

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

Available in IS, AH, CH

5.1.2.209 optixTransformVectorFromObjectToWorldSpace()

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

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

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

Available in IS, AH, CH

5.1.2.210 optixTransformVectorFromWorldToObjectSpace()

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

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

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

Available in IS, AH, CH

5.1.2.211 optixTraverse() [1/2]

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

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

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

Parameters

in	*type*	
in	*handle*	
in	*rayOrigin*	
in	*rayDirection*	
in	*tmin*	
in	*tmax*	
in	*rayTime*	
in	*visibilityMask*	really only 8 bits
in	*rayFlags*	really only 16 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

Available in RG, CH, MS, CC, DC

5.1.2.212 optixTraverse() [2/2]

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

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

Parameters

in	*handle*	

Parameters

in	*rayOrigin*	
in	*rayDirection*	
in	*tmin*	
in	*tmax*	
in	*rayTime*	
in	*visibilityMask*	really only 8 bits
in	*rayFlags*	really only 16 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

Available in RG, CH, MS, CC, DC

5.1.2.213 optixUndefinedValue()

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

Returns an undefined value.

Available anywhere

5.2 Function Table

Classes

- struct OptixFunctionTable

Typedefs

- typedef struct OptixFunctionTable OptixFunctionTable

Variables

- OptixFunctionTable g_optixFunctionTable

5.2.1 Detailed Description

OptiX Function Table.

5.2.2 Typedef Documentation

5.2.2.1 OptixFunctionTable

`typedef struct OptixFunctionTable OptixFunctionTable`

The function table containing all API functions.

See optixInit( ) and optixInitWithHandle( ).

5.2.3 Variable Documentation

5.2.3.1 g_optixFunctionTable

```
OptixFunctionTable g_optixFunctionTable
```

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

5.3 Host API

Modules

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

5.3.1 Detailed Description

OptiX Host API.

5.4 Error handling

5.5 Device context

5.6 Pipelines

5.7 Modules

5.8 Tasks

5.9 Program groups

5.10 Launches

5.11 Acceleration structures

5.12 Denoiser

5.13 Utilities

Classes

- struct OptixUtilDenoiserImageTile

Macros

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

Functions

- OPTIX_MICROMAP_INLINE_FUNC float optix_impl::__uint_as_float (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::extractEvenBits (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::prefixEor (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC void optix_impl::index2dbary (unsigned int index, unsigned int &u, unsigned int &v, unsigned int &w)
- OPTIX_MICROMAP_INLINE_FUNC void optix_impl::micro2bary (unsigned int index, unsigned int subdivisionLevel, float2 &bary0, float2 &bary1, float2 &bary2)
- OPTIX_MICROMAP_INLINE_FUNC float2 optix_impl::base2micro (const float2 &baseBarycentrics, const float2 microVertexBaseBarycentrics[3])
- OptixResult optixUtilGetPixelStride (const OptixImage2D &image, unsigned int &pixelStrideInBytes)
- OptixResult optixUtilDenoiserSplitImage (const OptixImage2D &input, const OptixImage2D &output, unsigned int overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight, std::vector< OptixUtilDenoiserImageTile > &tiles)
- OptixResult optixUtilDenoiserInvokeTiled (OptixDenoiser denoiser, CUstream stream, const OptixDenoiserParams *params, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, const OptixDenoiserGuideLayer *guideLayer, const OptixDenoiserLayer *layers, unsigned int numLayers, CUdeviceptr scratch, size_t scratchSizeInBytes, unsigned int overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight)
- OptixResult optixUtilAccumulateStackSizes (OptixProgramGroup programGroup, OptixStackSizes *stackSizes, OptixPipeline pipeline)
- OptixResult optixUtilComputeStackSizes (const OptixStackSizes *stackSizes, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepth, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesDCSplit (const OptixStackSizes *stackSizes, unsigned int dssDCFromTraversal, unsigned int dssDCFromState, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepthFromTraversal, unsigned int maxDCDepthFromState, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesCssCCTree (const OptixStackSizes *stackSizes, unsigned int cssCCTree, unsigned int maxTraceDepth, unsigned int maxDCDepth, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesSimplePathTracer (OptixProgramGroup programGroupRG, OptixProgramGroup programGroupMS1, const OptixProgramGroup *programGroupCH1, unsigned int programGroupCH1Count, OptixProgramGroup programGroupMS2, const OptixProgramGroup *programGroupCH2, unsigned int programGroupCH2Count, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize, OptixPipeline pipeline)
- OptixResult optixInitWithHandle (void **handlePtr)
- OptixResult optixInit (void)
- OptixResult optixUninitWithHandle (void *handle)

5.13.1 Detailed Description

OptiX Utilities.

5.13.2 Macro Definition Documentation

5.13.2.1 OPTIX_MICROMAP_FLOAT2_SUB

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

5.13.2.2 OPTIX_MICROMAP_INLINE_FUNC

```
#define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline
```

5.13.3 Function Documentation

5.13.3.1 __uint_as_float()

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

5.13.3.2 base2micro()

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

5.13.3.3 extractEvenBits()

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

5.13.3.4 index2dbary()

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

5.13.3.5 micro2bary()

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

5.13.3.6 optixInit()

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

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

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

5.13.3.7 optixInitWithHandle()

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

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

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

See also optixUninitWithHandle

5.13.3.8 optixUninitWithHandle()

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

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

See also optixInitWithHandle

5.13.3.9 optixUtilAccumulateStackSizes()

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

Retrieves direct and continuation stack sizes for each program in the program group and accumulates the upper bounds in the 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. If the programs rely on external functions, passing the current pipeline will consider these as well. Otherwise, a null pointer can be passed instead. When external functions are present, a warning will be issued for these cases.

5.13.3.10 optixUtilComputeStackSizes()

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

Computes the stack size values needed to configure a pipeline.

See the programming guide for an explanation of the formula.

Parameters

in	*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.

5.13.3.11 optixUtilComputeStackSizesCssCCTree( )

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

Computes the stack size values needed to configure a pipeline.

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

See programming guide for an explanation of the formula.

Parameters

in	*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.
out	*continuationStackSize*	Continuation stack requirement.

5.13.3.12 optixUtilComputeStackSizesDCSplit()

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

Computes the stack size values needed to configure a pipeline.

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

See programming guide for an explanation of the formula.

Parameters

in	*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.

5.13.3.13 optixUtilComputeStackSizesSimplePathTracer()

```
OptixResult optixUtilComputeStackSizesSimplePathTracer (
            OptixProgramGroup programGroupRG,
            OptixProgramGroup programGroupMS1,
            const OptixProgramGroup * programGroupCH1,
            unsigned int programGroupCH1Count,
```

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

Computes the stack size values needed to configure a pipeline.

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

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

See programming guide for an explanation of the formula.

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

5.13.3.14 optixUtilDenoiserInvokeTiled( )

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

Run denoiser on input layers see optixDenoiserInvoke additional parameters:

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

Parameters

in	*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*

5.13.3.15 optixUtilDenoiserSplitImage()

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

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

Parameters

in	*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

5.13.3.16 optixUtilGetPixelStride()

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

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

Parameters

in	*image*	Image containing the pixel stride
in	*pixelStrideInBytes*	Pixel stride in bytes

5.13.3.17 prefixEor()

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

5.14 Types

Classes

- struct OptixDeviceContextOptions
- struct OptixOpacityMicromapUsageCount
- struct OptixBuildInputOpacityMicromap
- struct OptixRelocateInputOpacityMicromap
- struct OptixDisplacementMicromapDesc
- struct OptixDisplacementMicromapHistogramEntry
- struct OptixDisplacementMicromapArrayBuildInput
- struct OptixDisplacementMicromapUsageCount
- struct OptixBuildInputDisplacementMicromap
- struct OptixBuildInputTriangleArray
- struct OptixRelocateInputTriangleArray
- struct OptixBuildInputCurveArray
- struct OptixBuildInputSphereArray
- struct OptixAabb
- struct OptixBuildInputCustomPrimitiveArray
- struct OptixBuildInputInstanceArray
- struct OptixRelocateInputInstanceArray
- struct OptixBuildInput
- struct OptixRelocateInput
- struct OptixInstance
- struct OptixOpacityMicromapDesc
- struct OptixOpacityMicromapHistogramEntry
- struct OptixOpacityMicromapArrayBuildInput
- struct OptixMicromapBufferSizes
- struct OptixMicromapBuffers
- struct OptixMotionOptions
- struct OptixAccelBuildOptions
- struct OptixAccelBufferSizes
- struct OptixAccelEmitDesc
- struct OptixRelocationInfo
- struct OptixStaticTransform
- struct OptixMatrixMotionTransform
- struct OptixSRTData
- struct OptixSRTMotionTransform
- struct OptixImage2D
- struct OptixDenoiserOptions

- struct OptixDenoiserGuideLayer
- struct OptixDenoiserLayer
- struct OptixDenoiserParams
- struct OptixDenoiserSizes
- struct OptixModuleCompileBoundValueEntry
- struct OptixPayloadType
- struct OptixModuleCompileOptions
- struct OptixProgramGroupSingleModule
- struct OptixProgramGroupHitgroup
- struct OptixProgramGroupCallables
- struct OptixProgramGroupDesc
- struct OptixProgramGroupOptions
- struct OptixPipelineCompileOptions
- struct OptixPipelineLinkOptions
- struct OptixShaderBindingTable
- struct OptixStackSizes
- struct 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_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
- #define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
- #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT (0)
- #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE (1)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT (2)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE (3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT (-1)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE (-2)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT (-3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE (-4)
- #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
- #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
- #define OPTIX_DISPLACEMENT_MICROMAP_MAX_SUBDIVISION_LEVEL 5
- #define OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
- #define OPTIX_DISPLACEMENT_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull

Typedefs

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

- typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags
- typedef enum OptixCurveEndcapFlags OptixCurveEndcapFlags
- typedef struct OptixBuildInputCurveArray OptixBuildInputCurveArray
- typedef struct OptixBuildInputSphereArray OptixBuildInputSphereArray
- typedef struct OptixAabb OptixAabb
- typedef struct OptixBuildInputCustomPrimitiveArray OptixBuildInputCustomPrimitiveArray
- typedef struct OptixBuildInputInstanceArray OptixBuildInputInstanceArray
- typedef struct OptixRelocateInputInstanceArray OptixRelocateInputInstanceArray
- typedef enum OptixBuildInputType OptixBuildInputType
- typedef struct OptixBuildInput OptixBuildInput
- typedef struct OptixRelocateInput OptixRelocateInput
- typedef enum OptixInstanceFlags OptixInstanceFlags
- typedef struct OptixInstance OptixInstance
- typedef enum OptixBuildFlags OptixBuildFlags
- typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags
- typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc
- typedef struct OptixOpacityMicromapHistogramEntry OptixOpacityMicromapHistogramEntry
- typedef struct OptixOpacityMicromapArrayBuildInput OptixOpacityMicromapArrayBuildInput
- typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
- typedef struct OptixMicromapBuffers OptixMicromapBuffers
- typedef enum OptixBuildOperation OptixBuildOperation
- typedef enum OptixMotionFlags OptixMotionFlags
- typedef struct OptixMotionOptions OptixMotionOptions
- typedef struct OptixAccelBuildOptions OptixAccelBuildOptions
- typedef struct OptixAccelBufferSizes OptixAccelBufferSizes
- typedef enum OptixAccelPropertyType OptixAccelPropertyType
- typedef struct OptixAccelEmitDesc OptixAccelEmitDesc
- typedef struct OptixRelocationInfo OptixRelocationInfo
- typedef struct OptixStaticTransform OptixStaticTransform
- typedef struct OptixMatrixMotionTransform OptixMatrixMotionTransform
- typedef struct OptixSRTData OptixSRTData
- typedef struct OptixSRTMotionTransform OptixSRTMotionTransform
- typedef enum OptixTraversableType OptixTraversableType
- typedef enum OptixPixelFormat OptixPixelFormat
- typedef struct OptixImage2D OptixImage2D
- typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
- typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
- typedef struct OptixDenoiserOptions OptixDenoiserOptions
- typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
- typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
- typedef struct OptixDenoiserLayer OptixDenoiserLayer
- typedef struct OptixDenoiserParams OptixDenoiserParams
- typedef struct OptixDenoiserSizes OptixDenoiserSizes
- typedef enum OptixRayFlags OptixRayFlags
- typedef enum OptixTransformType OptixTransformType
- typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags
- typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
- typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
- typedef enum OptixModuleCompileState OptixModuleCompileState
- typedef struct OptixModuleCompileBoundValueEntry OptixModuleCompileBoundValueEntry

- typedef enum OptixPayloadTypeID OptixPayloadTypeID
- typedef enum OptixPayloadSemantics OptixPayloadSemantics
- typedef struct OptixPayloadType OptixPayloadType
- typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
- typedef enum OptixProgramGroupKind OptixProgramGroupKind
- typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
- typedef struct OptixProgramGroupSingleModule OptixProgramGroupSingleModule
- typedef struct OptixProgramGroupHitgroup OptixProgramGroupHitgroup
- typedef struct OptixProgramGroupCallables OptixProgramGroupCallables
- typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
- typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
- typedef enum OptixExceptionCodes OptixExceptionCodes
- typedef enum OptixExceptionFlags OptixExceptionFlags
- typedef struct OptixPipelineCompileOptions OptixPipelineCompileOptions
- typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
- typedef struct OptixShaderBindingTable OptixShaderBindingTable
- typedef struct OptixStackSizes OptixStackSizes
- typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
- typedef OptixResult() OptixQueryFunctionTable_t(int abiId, unsigned int numOptions, OptixQueryFunctionTableOptions *, const void **, void *functionTable, size_t sizeOfTable)
- typedef struct 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_INPUT = 7200 ,
  OPTIX_ERROR_INVALID_LAUNCH_PARAMETER = 7201 ,
  OPTIX_ERROR_INVALID_PAYLOAD_ACCESS = 7202 ,
  OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS = 7203 ,
  OPTIX_ERROR_INVALID_FUNCTION_USE = 7204 ,
  OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS = 7205 ,
  OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250 ,
  OPTIX_ERROR_PIPELINE_LINK_ERROR = 7251 ,
  OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE = 7270 ,
  OPTIX_ERROR_INTERNAL_COMPILER_ERROR = 7299 ,
  OPTIX_ERROR_DENOISER_MODEL_NOT_SET = 7300 ,
  OPTIX_ERROR_DENOISER_NOT_INITIALIZED = 7301 ,
  OPTIX_ERROR_NOT_COMPATIBLE = 7400 ,

OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH = 7500 ,
OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED = 7501 ,
OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID = 7502 ,
OPTIX_ERROR_NOT_SUPPORTED = 7800 ,
OPTIX_ERROR_UNSUPPORTED_ABI_VERSION = 7801 ,
OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH = 7802 ,
OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS = 7803 ,
OPTIX_ERROR_LIBRARY_NOT_FOUND = 7804 ,
OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND = 7805 ,
OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE = 7806 ,
OPTIX_ERROR_DEVICE_OUT_OF_MEMORY = 7807 ,
OPTIX_ERROR_CUDA_ERROR = 7900 ,
OPTIX_ERROR_INTERNAL_ERROR = 7990 ,
OPTIX_ERROR_UNKNOWN = 7999 }

- enum OptixDeviceProperty {
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004 ,
  OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009 ,
  OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING = 0x200A }
- enum OptixDeviceContextValidationMode {
  OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0 ,
  OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF }
- enum OptixDevicePropertyShaderExecutionReorderingFlags {
  OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_NONE = 0 ,
  OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD = 1 << 0 }
- enum OptixGeometryFlags {
  OPTIX_GEOMETRY_FLAG_NONE = 0 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
  OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u << 1 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 2 }
- enum OptixHitKind {
  OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE ,
  OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF }
- enum OptixIndicesFormat {
  OPTIX_INDICES_FORMAT_NONE = 0 ,
  OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102 ,
  OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103 }
- enum OptixVertexFormat {
  OPTIX_VERTEX_FORMAT_NONE = 0 ,
  OPTIX_VERTEX_FORMAT_FLOAT3 = 0x2121 ,
  OPTIX_VERTEX_FORMAT_FLOAT2 = 0x2122 ,
  OPTIX_VERTEX_FORMAT_HALF3 = 0x2123 ,
  OPTIX_VERTEX_FORMAT_HALF2 = 0x2124 ,
  OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125 ,
  OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126 }

- enum OptixTransformFormat {
  OPTIX_TRANSFORM_FORMAT_NONE = 0 ,
  OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1 }
- enum OptixDisplacementMicromapBiasAndScaleFormat {
  OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_NONE = 0 ,
  OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_FLOAT2 = 0x2241 ,
  OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_HALF2 = 0x2242 }
- enum OptixDisplacementMicromapDirectionFormat {
  OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_NONE = 0 ,
  OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_FLOAT3 = 0x2261 ,
  OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_HALF3 = 0x2262 }
- enum OptixOpacityMicromapFormat {
  OPTIX_OPACITY_MICROMAP_FORMAT_NONE = 0 ,
  OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE = 1 ,
  OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE = 2 }
- enum OptixOpacityMicromapArrayIndexingMode {
  OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0 ,
  OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1 ,
  OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2 }
- enum OptixDisplacementMicromapFormat {
  OPTIX_DISPLACEMENT_MICROMAP_FORMAT_NONE = 0 ,
  OPTIX_DISPLACEMENT_MICROMAP_FORMAT_64_MICRO_TRIS_64_BYTES = 1 ,
  OPTIX_DISPLACEMENT_MICROMAP_FORMAT_256_MICRO_TRIS_128_BYTES = 2 ,
  OPTIX_DISPLACEMENT_MICROMAP_FORMAT_1024_MICRO_TRIS_128_BYTES = 3 }
- enum OptixDisplacementMicromapFlags {
  OPTIX_DISPLACEMENT_MICROMAP_FLAG_NONE = 0 ,
  OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 << 0 ,
  OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 << 1 }
- enum OptixDisplacementMicromapTriangleFlags {
  OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_NONE = 0 ,
  OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_01 = 1 << 0 ,
  OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_12 = 1 << 1 ,
  OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_20 = 1 << 2 }
- enum OptixDisplacementMicromapArrayIndexingMode {
  OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0 ,
  OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1 ,
  OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2 }
- enum OptixPrimitiveType {
  OPTIX_PRIMITIVE_TYPE_CUSTOM = 0x2500 ,
  OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE = 0x2501 ,
  OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE = 0x2502 ,
  OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR = 0x2503 ,
  OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM = 0x2504 ,
  OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE = 0x2505 ,
  OPTIX_PRIMITIVE_TYPE_SPHERE = 0x2506 ,
  OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER = 0x2507 ,
  OPTIX_PRIMITIVE_TYPE_TRIANGLE = 0x2531 ,
  OPTIX_PRIMITIVE_TYPE_DISPLACED_MICROMESH_TRIANGLE = 0x2532 }
- enum OptixPrimitiveTypeFlags {
  OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM = 1 << 0 ,
  OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE = 1 << 1 ,
  OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE = 1 << 2 ,
  OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR = 1 << 3 ,

OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM = 1 << 4 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE = 1 << 5 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE = 1 << 6 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER = 1 << 7 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE = 1 << 31 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_DISPLACED_MICROMESH_TRIANGLE = 1 << 30 }

- enum OptixCurveEndcapFlags {
  OPTIX_CURVE_ENDCAP_DEFAULT = 0 ,
  OPTIX_CURVE_ENDCAP_ON = 1 << 0 }
- enum OptixBuildInputType {
  OPTIX_BUILD_INPUT_TYPE_TRIANGLES = 0x2141 ,
  OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES = 0x2142 ,
  OPTIX_BUILD_INPUT_TYPE_INSTANCES = 0x2143 ,
  OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS = 0x2144 ,
  OPTIX_BUILD_INPUT_TYPE_CURVES = 0x2145 ,
  OPTIX_BUILD_INPUT_TYPE_SPHERES = 0x2146 }
- enum OptixInstanceFlags {
  OPTIX_INSTANCE_FLAG_NONE = 0 ,
  OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 0 ,
  OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u << 1 ,
  OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT = 1u << 2 ,
  OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u << 3 ,
  OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 4 ,
  OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS = 1u << 5 }
- enum OptixBuildFlags {
  OPTIX_BUILD_FLAG_NONE = 0 ,
  OPTIX_BUILD_FLAG_ALLOW_UPDATE = 1u << 0 ,
  OPTIX_BUILD_FLAG_ALLOW_COMPACTION = 1u << 1 ,
  OPTIX_BUILD_FLAG_PREFER_FAST_TRACE = 1u << 2 ,
  OPTIX_BUILD_FLAG_PREFER_FAST_BUILD = 1u << 3 ,
  OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS = 1u << 4 ,
  OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS = 1u << 5 ,
  OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE = 1u << 6 ,
  OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1u << 7 }
- enum OptixOpacityMicromapFlags {
  OPTIX_OPACITY_MICROMAP_FLAG_NONE = 0 ,
  OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 << 0 ,
  OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 << 1 }
- enum OptixBuildOperation {
  OPTIX_BUILD_OPERATION_BUILD = 0x2161 ,
  OPTIX_BUILD_OPERATION_UPDATE = 0x2162 }
- enum OptixMotionFlags {
  OPTIX_MOTION_FLAG_NONE = 0 ,
  OPTIX_MOTION_FLAG_START_VANISH = 1u << 0 ,
  OPTIX_MOTION_FLAG_END_VANISH = 1u << 1 }
- enum OptixAccelPropertyType {
  OPTIX_PROPERTY_TYPE_COMPACTED_SIZE = 0x2181 ,
  OPTIX_PROPERTY_TYPE_AABBS = 0x2182 }
- enum OptixTraversableType {
  OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM = 0x21C1 ,
  OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM = 0x21C2 ,
  OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM = 0x21C3 }

- enum OptixPixelFormat {
  OPTIX_PIXEL_FORMAT_HALF1 = 0x220a ,
  OPTIX_PIXEL_FORMAT_HALF2 = 0x2207 ,
  OPTIX_PIXEL_FORMAT_HALF3 = 0x2201 ,
  OPTIX_PIXEL_FORMAT_HALF4 = 0x2202 ,
  OPTIX_PIXEL_FORMAT_FLOAT1 = 0x220b ,
  OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208 ,
  OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203 ,
  OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204 ,
  OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205 ,
  OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206 ,
  OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209 }
- enum OptixDenoiserModelKind {
  OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322 ,
  OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323 ,
  OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326 ,
  OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328 }
- enum OptixDenoiserAlphaMode {
  OPTIX_DENOISER_ALPHA_MODE_COPY = 0 ,
  OPTIX_DENOISER_ALPHA_MODE_DENOISE = 1 }
- enum OptixDenoiserAOVType {
  OPTIX_DENOISER_AOV_TYPE_NONE = 0 ,
  OPTIX_DENOISER_AOV_TYPE_BEAUTY = 0x7000 ,
  OPTIX_DENOISER_AOV_TYPE_SPECULAR = 0x7001 ,
  OPTIX_DENOISER_AOV_TYPE_REFLECTION = 0x7002 ,
  OPTIX_DENOISER_AOV_TYPE_REFRACTION = 0x7003 ,
  OPTIX_DENOISER_AOV_TYPE_DIFFUSE = 0x7004 }
- enum OptixRayFlags {
  OPTIX_RAY_FLAG_NONE = 0u ,
  OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
  OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u << 1 ,
  OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u << 2 ,
  OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u << 3 ,
  OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u << 4 ,
  OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u << 5 ,
  OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u << 6 ,
  OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u << 7 ,
  OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 10 }
- enum OptixTransformType {
  OPTIX_TRANSFORM_TYPE_NONE = 0 ,
  OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM = 1 ,
  OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM = 2 ,
  OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM = 3 ,
  OPTIX_TRANSFORM_TYPE_INSTANCE = 4 }
- enum OptixTraversableGraphFlags {
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u << 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u << 1 }
- enum OptixCompileOptimizationLevel {
  OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0 ,

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

- enum OptixCompileDebugLevel {
  OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT = 0 ,
  OPTIX_COMPILE_DEBUG_LEVEL_NONE = 0x2350 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL = 0x2351 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353 ,
  OPTIX_COMPILE_DEBUG_LEVEL_FULL = 0x2352 }
- enum OptixModuleCompileState {
  OPTIX_MODULE_COMPILE_STATE_NOT_STARTED = 0x2360 ,
  OPTIX_MODULE_COMPILE_STATE_STARTED = 0x2361 ,
  OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE = 0x2362 ,
  OPTIX_MODULE_COMPILE_STATE_FAILED = 0x2363 ,
  OPTIX_MODULE_COMPILE_STATE_COMPLETED = 0x2364 }
- enum OptixPayloadTypeID {
  OPTIX_PAYLOAD_TYPE_DEFAULT = 0 ,
  OPTIX_PAYLOAD_TYPE_ID_0 = (1 << 0u) ,
  OPTIX_PAYLOAD_TYPE_ID_1 = (1 << 1u) ,
  OPTIX_PAYLOAD_TYPE_ID_2 = (1 << 2u) ,
  OPTIX_PAYLOAD_TYPE_ID_3 = (1 << 3u) ,
  OPTIX_PAYLOAD_TYPE_ID_4 = (1 << 4u) ,
  OPTIX_PAYLOAD_TYPE_ID_5 = (1 << 5u) ,
  OPTIX_PAYLOAD_TYPE_ID_6 = (1 << 6u) ,
  OPTIX_PAYLOAD_TYPE_ID_7 = (1 << 7u) }
- enum OptixPayloadSemantics {
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ = 1u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE = 2u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_READ = 1u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_WRITE = 2u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE = 3u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_READ = 1u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_WRITE = 2u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE = 3u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_READ = 1u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_WRITE = 2u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE = 3u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_READ = 1u << 8 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_WRITE = 2u << 8 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE = 3u << 8 }
- enum OptixProgramGroupKind {
  OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421 ,
  OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422 ,
  OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423 ,
  OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424 ,
  OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425 }

- enum OptixProgramGroupFlags { OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0 }
- enum OptixExceptionCodes {
  OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1 ,
  OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2 }
- enum OptixExceptionFlags {
  OPTIX_EXCEPTION_FLAG_NONE = 0 ,
  OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u << 0 ,
  OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u << 1 ,
  OPTIX_EXCEPTION_FLAG_USER = 1u << 2 }
- enum OptixQueryFunctionTableOptions { OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0 }

5.14.1 Detailed Description

OptiX Types.

5.14.2 Macro Definition Documentation

5.14.2.1 OPTIX_AABB_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8ull
```

Alignment requirement for OptixBuildInputCustomPrimitiveArray::aabbBuffers.

5.14.2.2 OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128ull
```

Alignment requirement for output and temporay buffers for acceleration structures.

5.14.2.3 OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
```

Maximum number of payload types allowed.

5.14.2.4 OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
```

Maximum number of payload values allowed.

5.14.2.5 OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
```

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

5.14.2.6 OPTIX_DISPLACEMENT_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_DISPLACEMENT_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
```

Alignment requirement for displacement micromap array buffers.

5.14.2.7 OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
```

Alignment requirement for displacement micromap descriptor buffers.

5.14.2.8 OPTIX_DISPLACEMENT_MICROMAP_MAX_SUBDIVISION_LEVEL

```
#define OPTIX_DISPLACEMENT_MICROMAP_MAX_SUBDIVISION_LEVEL 5
```

Maximum subdivision level for displacement micromaps.

5.14.2.9 OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT

```
#define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16ull
```

Alignment requirement for OptixBuildInputTriangleArray::preTransform.

5.14.2.10 OPTIX_INSTANCE_BYTE_ALIGNMENT

```
#define OPTIX_INSTANCE_BYTE_ALIGNMENT 16ull
```

Alignment requirement for OptixBuildInputInstanceArray::instances.

5.14.2.11 OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
```

Alignment requirement for opacity micromap array buffers.

5.14.2.12 OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
```

Alignment requirement for OptixOpacityMicromapArrayBuildInput::perMicromapDescBuffer.

5.14.2.13 OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL

```
#define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
```

Maximum subdivision level for opacity micromaps.

5.14.2.14 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE

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

5.14.2.15 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT

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

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

5.14.2.16 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE

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

5.14.2.17 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPAREN

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

5.14.2.18 OPTIX_OPACITY_MICROMAP_STATE_OPAQUE

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

5.14.2.19 OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT

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

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

5.14.2.20 OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE

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

5.14.2.21 OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT

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

5.14.2.22 OPTIX_SBT_RECORD_ALIGNMENT

```
#define OPTIX_SBT_RECORD_ALIGNMENT 16ull
```

Alignment requirement for device pointers in OptixShaderBindingTable.

5.14.2.23 OPTIX_SBT_RECORD_HEADER_SIZE

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

Size of the SBT record headers.

5.14.2.24 OPTIX_TRANSFORM_BYTE_ALIGNMENT

```
#define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
```

Alignment requirement for OptixStaticTransform, OptixMatrixMotionTransform, OptixSRTMotionTransform.

5.14.3 Typedef Documentation

5.14.3.1 CUdeviceptr

```
typedef unsigned long long CUdeviceptr
```

CUDA device pointer.

5.14.3.2 OptixAabb

```
typedef struct OptixAabb OptixAabb
```

AABB inputs.

5.14.3.3 OptixAccelBufferSizes

```
typedef struct OptixAccelBufferSizes OptixAccelBufferSizes
```

Struct for querying builder allocation requirements.

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

See also optixAccelComputeMemoryUsage( )

5.14.3.4 OptixAccelBuildOptions

```
typedef struct OptixAccelBuildOptions OptixAccelBuildOptions
```

Build options for acceleration structures.

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( )

5.14.3.5 OptixAccelEmitDesc

`typedef struct OptixAccelEmitDesc OptixAccelEmitDesc`

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

See also optixAccelBuild( )

5.14.3.6 OptixAccelPropertyType

`typedef enum OptixAccelPropertyType OptixAccelPropertyType`

Properties which can be emitted during acceleration structure build.

See also OptixAccelEmitDesc::type.

5.14.3.7 OptixBuildFlags

`typedef enum OptixBuildFlags OptixBuildFlags`

Builder Options.

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

5.14.3.8 OptixBuildInput

`typedef struct OptixBuildInput OptixBuildInput`

Build inputs.

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

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( )

5.14.3.9 OptixBuildInputCurveArray

`typedef struct OptixBuildInputCurveArray OptixBuildInputCurveArray`

Curve inputs.

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

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

5.14.3.10 OptixBuildInputCustomPrimitiveArray

`typedef struct OptixBuildInputCustomPrimitiveArray`

`OptixBuildInputCustomPrimitiveArray`

Custom primitive inputs.

See also OptixBuildInput::customPrimitiveArray

5.14.3.11 OptixBuildInputDisplacementMicromap

```
typedef struct OptixBuildInputDisplacementMicromap
OptixBuildInputDisplacementMicromap
```

Optional displacement part of a triangle array input.

5.14.3.12 OptixBuildInputInstanceArray

```
typedef struct OptixBuildInputInstanceArray OptixBuildInputInstanceArray
```

Instance and instance pointer inputs.

See also OptixBuildInput::instanceArray

5.14.3.13 OptixBuildInputOpacityMicromap

```
typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap
```

5.14.3.14 OptixBuildInputSphereArray

```
typedef struct OptixBuildInputSphereArray OptixBuildInputSphereArray
```

Sphere inputs.

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

The vertex buffers and radius buffers point to a host array of device pointers, one per motion step. Host array size must match the number of motion keys as set in OptixMotionOptions (or an array of size 1 if OptixMotionOptions::numKeys is set to 0 or 1). Each per motion key device pointer must point to an array of vertices corresponding to the center points of the spheres, or an array of 1 or N radii. Format OPTIX_VERTEX_FORMAT_FLOAT3 is used for vertices, OPTIX_VERTEX_FORMAT_FLOAT for radii.

See also OptixBuildInput::sphereArray

5.14.3.15 OptixBuildInputTriangleArray

```
typedef struct OptixBuildInputTriangleArray OptixBuildInputTriangleArray
```

Triangle inputs.

See also OptixBuildInput::triangleArray

5.14.3.16 OptixBuildInputType

```
typedef enum OptixBuildInputType OptixBuildInputType
```

Enum to distinguish the different build input types.

See also OptixBuildInput::type

5.14.3.17 OptixBuildOperation

```
typedef enum OptixBuildOperation OptixBuildOperation
```

Enum to specify the acceleration build operation.

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

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

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( ), OptixAccelBuildOptions

5.14.3.18 OptixBuiltinISOptions

```
typedef struct OptixBuiltinISOptions OptixBuiltinISOptions
```

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

See also optixBuiltinISModuleGet( )

5.14.3.19 OptixCompileDebugLevel

```
typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
```

Debug levels.

See also OptixModuleCompileOptions::debugLevel

5.14.3.20 OptixCompileOptimizationLevel

```
typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
```

Optimization levels.

See also OptixModuleCompileOptions::optLevel

5.14.3.21 OptixCurveEndcapFlags

```
typedef enum OptixCurveEndcapFlags OptixCurveEndcapFlags
```

Curve end cap types, for non-linear curves.

5.14.3.22 OptixDenoiser

```
typedef struct OptixDenoiser_t* OptixDenoiser
```

Opaque type representing a denoiser instance.

5.14.3.23 OptixDenoiserAlphaMode

```
typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
```

Alpha denoising mode.

See also optixDenoiserCreate( )

5.14.3.24 OptixDenoiserAOVType

```
typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
```

AOV type used by the denoiser.

5.14.3.25 OptixDenoiserGuideLayer

```
typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
```

Guide layer for the denoiser.

See also optixDenoiserInvoke( )

5.14.3.26 OptixDenoiserLayer

```
typedef struct OptixDenoiserLayer OptixDenoiserLayer
```

Input/Output layers for the denoiser.

See also optixDenoiserInvoke( )

5.14.3.27 OptixDenoiserModelKind

```
typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
```

Model kind used by the denoiser.

See also optixDenoiserCreate

5.14.3.28 OptixDenoiserOptions

```
typedef struct OptixDenoiserOptions OptixDenoiserOptions
```

Options used by the denoiser.

See also optixDenoiserCreate( )

5.14.3.29 OptixDenoiserParams

```
typedef struct OptixDenoiserParams OptixDenoiserParams
```

Various parameters used by the denoiser.

See also optixDenoiserInvoke( )

optixDenoiserComputeIntensity( )

optixDenoiserComputeAverageColor( )

5.14.3.30 OptixDenoiserSizes

```
typedef struct OptixDenoiserSizes OptixDenoiserSizes
```

Various sizes related to the denoiser.

See also optixDenoiserComputeMemoryResources( )

5.14.3.31 OptixDeviceContext

```
typedef struct OptixDeviceContext_t* OptixDeviceContext
```

Opaque type representing a device context.

5.14.3.32 OptixDeviceContextOptions

```
typedef struct OptixDeviceContextOptions OptixDeviceContextOptions
```

Parameters used for optixDeviceContextCreate( )

See also optixDeviceContextCreate( )

5.14.3.33 OptixDeviceContextValidationMode

`typedef enum OptixDeviceContextValidationMode OptixDeviceContextValidationMode`

Validation mode settings.

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

See also optixDeviceContextCreate( )

5.14.3.34 OptixDeviceProperty

`typedef enum OptixDeviceProperty OptixDeviceProperty`

Parameters used for optixDeviceContextGetProperty( )

See also optixDeviceContextGetProperty( )

5.14.3.35 OptixDevicePropertyShaderExecutionReorderingFlags

`typedef enum OptixDevicePropertyShaderExecutionReorderingFlags OptixDevicePropertyShaderExecutionReorderingFlags`

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

See also optixDeviceContextGetProperty( )

5.14.3.36 OptixDisplacementMicromapArrayBuildInput

`typedef struct OptixDisplacementMicromapArrayBuildInput OptixDisplacementMicromapArrayBuildInput`

Inputs to displacement micromaps array construction.

5.14.3.37 OptixDisplacementMicromapArrayIndexingMode

`typedef enum OptixDisplacementMicromapArrayIndexingMode OptixDisplacementMicromapArrayIndexingMode`

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

5.14.3.38 OptixDisplacementMicromapBiasAndScaleFormat

`typedef enum OptixDisplacementMicromapBiasAndScaleFormat OptixDisplacementMicromapBiasAndScaleFormat`

5.14.3.39 OptixDisplacementMicromapDesc

`typedef struct OptixDisplacementMicromapDesc OptixDisplacementMicromapDesc`

5.14.3.40 OptixDisplacementMicromapDirectionFormat

`typedef enum OptixDisplacementMicromapDirectionFormat OptixDisplacementMicromapDirectionFormat`

5.14.3.41 OptixDisplacementMicromapFlags

```
typedef enum OptixDisplacementMicromapFlags OptixDisplacementMicromapFlags
```

Flags defining behavior of DMMs in a DMM array.

5.14.3.42 OptixDisplacementMicromapFormat

```
typedef enum OptixDisplacementMicromapFormat OptixDisplacementMicromapFormat
```

DMM input data format.

5.14.3.43 OptixDisplacementMicromapHistogramEntry

```
typedef struct OptixDisplacementMicromapHistogramEntry
OptixDisplacementMicromapHistogramEntry
```

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

5.14.3.44 OptixDisplacementMicromapTriangleFlags

```
typedef enum OptixDisplacementMicromapTriangleFlags
OptixDisplacementMicromapTriangleFlags
```

5.14.3.45 OptixDisplacementMicromapUsageCount

```
typedef struct OptixDisplacementMicromapUsageCount
OptixDisplacementMicromapUsageCount
```

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

5.14.3.46 OptixExceptionCodes

```
typedef enum OptixExceptionCodes OptixExceptionCodes
```

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

5.14.3.47 OptixExceptionFlags

```
typedef enum OptixExceptionFlags OptixExceptionFlags
```

Exception flags.

See also OptixPipelineCompileOptions::exceptionFlags, OptixExceptionCodes

5.14.3.48 OptixGeometryFlags

```
typedef enum OptixGeometryFlags OptixGeometryFlags
```

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

5.14.3.49 OptixHitKind

```
typedef enum OptixHitKind OptixHitKind
```

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

See also optixGetHitKind( )

5.14.3.50 OptixImage2D

```
typedef struct OptixImage2D OptixImage2D
```

Image descriptor used by the denoiser.

See also optixDenoiserInvoke( ), optixDenoiserComputeIntensity( )

5.14.3.51 OptixIndicesFormat

```
typedef enum OptixIndicesFormat OptixIndicesFormat
```

Format of indices used int OptixBuildInputTriangleArray::indexFormat.

5.14.3.52 OptixInstance

```
typedef struct OptixInstance OptixInstance
```

Instances.

See also OptixBuildInputInstanceArray::instances

5.14.3.53 OptixInstanceFlags

```
typedef enum OptixInstanceFlags OptixInstanceFlags
```

Flags set on the OptixInstance::flags.

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

5.14.3.54 OptixLogCallback

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

Type of the callback function used for log messages.

Parameters

in	*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

5.14.3.55 OptixMatrixMotionTransform

```
typedef struct OptixMatrixMotionTransform OptixMatrixMotionTransform
```

Represents a matrix motion transformation.

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

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

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

See also optixConvertPointerToTraversableHandle()

5.14.3.56 OptixMicromapBuffers

```
typedef struct OptixMicromapBuffers OptixMicromapBuffers
```

Buffer inputs for opacity/displacement micromap array builds.

5.14.3.57 OptixMicromapBufferSizes

```
typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
```

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

5.14.3.58 OptixModule

```
typedef struct OptixModule_t* OptixModule
```

Opaque type representing a module.

5.14.3.59 OptixModuleCompileBoundValueEntry

```
typedef struct OptixModuleCompileBoundValueEntry
OptixModuleCompileBoundValueEntry
```

Struct for specifying specializations for pipelineParams as specified in OptixPipelineCompileOptions::pipelineLaunchParamsVariableName.

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

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

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

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

See also OptixModuleCompileOptions

5.14.3.60 OptixModuleCompileOptions

```
typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
```

Compilation options for module.

See also optixModuleCreate( )

5.14.3.61 OptixModuleCompileState

```
typedef enum OptixModuleCompileState OptixModuleCompileState
```

Module compilation state.

See also optixModuleGetCompilationState( ), optixModuleCreateWithTasks( )

5.14.3.62 OptixMotionFlags

```
typedef enum OptixMotionFlags OptixMotionFlags
```

Enum to specify motion flags.

See also OptixMotionOptions::flags.

5.14.3.63 OptixMotionOptions

```
typedef struct OptixMotionOptions OptixMotionOptions
```

Motion options.

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

5.14.3.64 OptixOpacityMicromapArrayBuildInput

```
typedef struct OptixOpacityMicromapArrayBuildInput
OptixOpacityMicromapArrayBuildInput
```

Inputs to opacity micromap array construction.

5.14.3.65 OptixOpacityMicromapArrayIndexingMode

```
typedef enum OptixOpacityMicromapArrayIndexingMode
OptixOpacityMicromapArrayIndexingMode
```

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

5.14.3.66 OptixOpacityMicromapDesc

`typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc`

Opacity micromap descriptor.

5.14.3.67 OptixOpacityMicromapFlags

`typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags`

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

5.14.3.68 OptixOpacityMicromapFormat

`typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat`

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

5.14.3.69 OptixOpacityMicromapHistogramEntry

`typedef struct OptixOpacityMicromapHistogramEntry OptixOpacityMicromapHistogramEntry`

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

5.14.3.70 OptixOpacityMicromapUsageCount

`typedef struct OptixOpacityMicromapUsageCount OptixOpacityMicromapUsageCount`

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

5.14.3.71 OptixPayloadSemantics

`typedef enum OptixPayloadSemantics OptixPayloadSemantics`

Semantic flags for a single payload word.

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

"trace_caller_write": Shaders may consume the value of the payload word passed to optixTrace by the caller. "trace_caller_read": The caller to optixTrace may read the payload word after the call to optixTrace.

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

5.14.3.72 OptixPayloadType

`typedef struct OptixPayloadType OptixPayloadType`

Specifies a single payload type.

5.14.3.73 OptixPayloadTypeID

`typedef enum OptixPayloadTypeID OptixPayloadTypeID`

Payload type identifiers.

5.14.3.74 OptixPipeline

`typedef struct OptixPipeline_t* OptixPipeline`

Opaque type representing a pipeline.

5.14.3.75 OptixPipelineCompileOptions

`typedef struct OptixPipelineCompileOptions OptixPipelineCompileOptions`

Compilation options for all modules of a pipeline.

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

See also optixModuleCreate( ), optixPipelineCreate( )

5.14.3.76 OptixPipelineLinkOptions

`typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions`

Link options for a pipeline.

See also optixPipelineCreate( )

5.14.3.77 OptixPixelFormat

`typedef enum OptixPixelFormat OptixPixelFormat`

Pixel formats used by the denoiser.

See also OptixImage2D::format

5.14.3.78 OptixPrimitiveType

`typedef enum OptixPrimitiveType OptixPrimitiveType`

Builtin primitive types.

5.14.3.79 OptixPrimitiveTypeFlags

`typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags`

Builtin flags may be bitwise combined.

See also OptixPipelineCompileOptions::usesPrimitiveTypeFlags

5.14.3.80 OptixProgramGroup

`typedef struct OptixProgramGroup_t* OptixProgramGroup`

Opaque type representing a program group.

5.14.3.81 OptixProgramGroupCallables

```
typedef struct OptixProgramGroupCallables OptixProgramGroupCallables
```

Program group representing callables.

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

See also #OptixProgramGroupDesc::callables

5.14.3.82 OptixProgramGroupDesc

```
typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
```

Descriptor for program groups.

5.14.3.83 OptixProgramGroupFlags

```
typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
```

Flags for program groups.

5.14.3.84 OptixProgramGroupHitgroup

```
typedef struct OptixProgramGroupHitgroup OptixProgramGroupHitgroup
```

Program group representing the hitgroup.

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

See also OptixProgramGroupDesc::hitgroup

5.14.3.85 OptixProgramGroupKind

```
typedef enum OptixProgramGroupKind OptixProgramGroupKind
```

Distinguishes different kinds of program groups.

5.14.3.86 OptixProgramGroupOptions

```
typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
```

Program group options.

See also optixProgramGroupCreate( )

5.14.3.87 OptixProgramGroupSingleModule

```
typedef struct OptixProgramGroupSingleModule OptixProgramGroupSingleModule
```

Program group representing a single module.

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

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

5.14.3.88 OptixQueryFunctionTable_t

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

*functionTable, size_t sizeOfTable)

Type of the function optixQueryFunctionTable( )

5.14.3.89 OptixQueryFunctionTableOptions

typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions

Options that can be passed to optixQueryFunctionTable( )

5.14.3.90 OptixRayFlags

typedef enum OptixRayFlags OptixRayFlags

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

See also optixTrace( )

5.14.3.91 OptixRelocateInput

typedef struct OptixRelocateInput OptixRelocateInput

Relocation inputs.

See also optixAccelRelocate( )

5.14.3.92 OptixRelocateInputInstanceArray

typedef struct OptixRelocateInputInstanceArray
OptixRelocateInputInstanceArray

Instance and instance pointer inputs.

See also OptixRelocateInput::instanceArray

5.14.3.93 OptixRelocateInputOpacityMicromap

typedef struct OptixRelocateInputOpacityMicromap
OptixRelocateInputOpacityMicromap

5.14.3.94 OptixRelocateInputTriangleArray

typedef struct OptixRelocateInputTriangleArray
OptixRelocateInputTriangleArray

Triangle inputs.

See also OptixRelocateInput::triangleArray

5.14.3.95 OptixRelocationInfo

typedef struct OptixRelocationInfo OptixRelocationInfo

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

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

5.14.3.96 OptixResult

typedef enum OptixResult OptixResult

Result codes returned from API functions.

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

See also optixGetErrorName( ), optixGetErrorString( )

5.14.3.97 OptixShaderBindingTable

```
typedef struct OptixShaderBindingTable OptixShaderBindingTable
```

Describes the shader binding table (SBT)

See also optixLaunch( )

5.14.3.98 OptixSRTData

```
typedef struct OptixSRTData OptixSRTData
```

Represents an SRT transformation.

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

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

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

The translation matrix $T = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \end{bmatrix}$ defines another translation that is applied after the rotation. Typically, this translation includes the inverse translation from the matrix S to reverse the translation for the pivot point for R.

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

See also OptixSRTMotionTransform::srtData, optixConvertPointerToTraversableHandle( )

5.14.3.99 OptixSRTMotionTransform

```
typedef struct OptixSRTMotionTransform OptixSRTMotionTransform
```

Represents an SRT motion transformation.

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

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

```
OptixSRTData srtData[N];
```

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

See also optixConvertPointerToTraversableHandle( )

5.14.3.100 OptixStackSizes

`typedef struct OptixStackSizes OptixStackSizes`

Describes the stack size requirements of a program group.

See also optixProgramGroupGetStackSize( )

5.14.3.101 OptixStaticTransform

`typedef struct OptixStaticTransform OptixStaticTransform`

Static transform.

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

See also optixConvertPointerToTraversableHandle( )

5.14.3.102 OptixTask

`typedef struct OptixTask_t* OptixTask`

Opaque type representing a work task.

5.14.3.103 OptixTransformFormat

`typedef enum OptixTransformFormat OptixTransformFormat`

Format of transform used in OptixBuildInputTriangleArray::transformFormat.

5.14.3.104 OptixTransformType

`typedef enum OptixTransformType OptixTransformType`

Transform.

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

5.14.3.105 OptixTraversableGraphFlags

`typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags`

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

5.14.3.106 OptixTraversableHandle

`typedef unsigned long long OptixTraversableHandle`

Traversable handle.

5.14.3.107 OptixTraversableType

`typedef enum OptixTraversableType OptixTraversableType`

Traversable Handles.

See also optixConvertPointerToTraversableHandle( )

5.14.3.108 OptixVertexFormat

`typedef enum OptixVertexFormat OptixVertexFormat`

Format of vertices used in OptixBuildInputTriangleArray::vertexFormat.

5.14.3.109 OptixVisibilityMask

`typedef unsigned int OptixVisibilityMask`

Visibility mask.

5.14.4 Enumeration Type Documentation

5.14.4.1 OptixAccelPropertyType

`enum OptixAccelPropertyType`

Properties which can be emitted during acceleration structure build.

See also OptixAccelEmitDesc::type.

Enumerator

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

5.14.4.2 OptixBuildFlags

`enum OptixBuildFlags`

Builder Options.

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

Enumerator

OPTIX_BUILD_FLAG_NONE	No special flags set.
OPTIX_BUILD_FLAG_ALLOW_UPDATE	Allow updating the build with new vertex positions with subsequent calls to optixAccelBuild.
OPTIX_BUILD_FLAG_ALLOW_COMPACTION	
OPTIX_BUILD_FLAG_PREFER_FAST_TRACE	This flag is mutually exclusive with OPTIX_BUILD_FLAG_PREFER_FAST_BUILD.

Enumerator

OPTIX_BUILD_FLAG_PREFER_FAST_BUILD	This flag is mutually exclusive with OPTIX_BUILD_FLAG_PREFER_FAST_TRACE.
OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS	Allow random access to build input vertices See optixGetTriangleVertexData optixGetLinearCurveVertexData optixGetQuadraticBSplineVertexData optixGetCubicBSplineVertexData optixGetCatmullRomVertexData optixGetRibbonVertexData optixGetRibbonNormal 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.
OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS	If enabled, any instances referencing this GAS are allowed to disable the opacity micromap test through the DISABLE_OPACITY_MICROMAPS flag instance flag. Note that the GAS will not be optimized for the attached opacity micromap Arrays if this flag is set, which may result in reduced traversal performance.

5.14.4.3 OptixBuildInputType

`enum OptixBuildInputType`

Enum to distinguish the different build input types.

See also OptixBuildInput::type

Enumerator

OPTIX_BUILD_INPUT_TYPE_TRIANGLES	Triangle inputs. See also 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

5.14.4.4 OptixBuildOperation

`enum OptixBuildOperation`

Enum to specify the acceleration build operation.

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

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

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( ), OptixAccelBuildOptions

Enumerator

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

5.14.4.5 OptixCompileDebugLevel

`enum OptixCompileDebugLevel`

Debug levels.

See also OptixModuleCompileOptions::debugLevel

Enumerator

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

5.14.4.6 OptixCompileOptimizationLevel

`enum OptixCompileOptimizationLevel`

Optimization levels.

See also OptixModuleCompileOptions::optLevel

Enumerator

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

Enumerator

OPTIX_COMPILE_OPTIMIZATION_LEVEL_3	All optimizations.

5.14.4.7 OptixCurveEndcapFlags

`enum OptixCurveEndcapFlags`

Curve end cap types, for non-linear curves.

Enumerator

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

5.14.4.8 OptixDenoiserAlphaMode

`enum OptixDenoiserAlphaMode`

Alpha denoising mode.

See also optixDenoiserCreate( )

Enumerator

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

5.14.4.9 OptixDenoiserAOVType

`enum OptixDenoiserAOVType`

AOV type used by the denoiser.

Enumerator

OPTIX_DENOISER_AOV_TYPE_NONE	Unspecified AOV type.
OPTIX_DENOISER_AOV_TYPE_BEAUTY	
OPTIX_DENOISER_AOV_TYPE_SPECULAR	
OPTIX_DENOISER_AOV_TYPE_REFLECTION	
OPTIX_DENOISER_AOV_TYPE_REFRACTION	
OPTIX_DENOISER_AOV_TYPE_DIFFUSE	

5.14.4.10 OptixDenoiserModelKind

`enum OptixDenoiserModelKind`

Model kind used by the denoiser.

See also optixDenoiserCreate

Enumerator

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

5.14.4.11 OptixDeviceContextValidationMode

`enum OptixDeviceContextValidationMode`

Validation mode settings.

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

See also optixDeviceContextCreate( )

Enumerator

OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF
OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL

5.14.4.12 OptixDeviceProperty

`enum OptixDeviceProperty`

Parameters used for optixDeviceContextGetProperty( )

See also optixDeviceContextGetProperty( )

Enumerator

OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH	Maximum value for OptixPipelineLinkOptions ::maxTraceDepth. sizeof(unsigned int)

Enumerator

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 summed value of OptixInstance ::sbtOffset. Also the maximum summed value of sbt offsets of all ancestor instances of a GAS in a traversable graph. sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING	Returns a flag specifying capabilities of the optixReorder( ) device function. See OptixDevicePropertyShaderExecutionReorderingFlags for documentation on the values that can be returned. sizeof(unsigned int)

5.14.4.13 OptixDevicePropertyShaderExecutionReorderingFlags

`enum OptixDevicePropertyShaderExecutionReorderingFlags`

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

See also optixDeviceContextGetProperty( )

Enumerator

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

Enumerator

OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD	

5.14.4.14 OptixDisplacementMicromapArrayIndexingMode

`enum OptixDisplacementMicromapArrayIndexingMode`

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

Enumerator

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

5.14.4.15 OptixDisplacementMicromapBiasAndScaleFormat

`enum OptixDisplacementMicromapBiasAndScaleFormat`

Enumerator

OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_NONE
OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_FLOAT2
OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_HALF2

5.14.4.16 OptixDisplacementMicromapDirectionFormat

`enum OptixDisplacementMicromapDirectionFormat`

Enumerator

OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_NONE
OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_FLOAT3
OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_HALF3

5.14.4.17 OptixDisplacementMicromapFlags

`enum OptixDisplacementMicromapFlags`

Flags defining behavior of DMMs in a DMM array.

Enumerator

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

5.14.4.18 OptixDisplacementMicromapFormat

`enum OptixDisplacementMicromapFormat`

DMM input data format.

Enumerator

OPTIX_DISPLACEMENT_MICROMAP_FORMAT_NONE
OPTIX_DISPLACEMENT_MICROMAP_FORMAT_64_MICRO_TRIS_64_BYTES
OPTIX_DISPLACEMENT_MICROMAP_FORMAT_256_MICRO_TRIS_128_BYTES
OPTIX_DISPLACEMENT_MICROMAP_FORMAT_1024_MICRO_TRIS_128_BYTES

5.14.4.19 OptixDisplacementMicromapTriangleFlags

`enum OptixDisplacementMicromapTriangleFlags`

Enumerator

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

5.14.4.20 OptixExceptionCodes

`enum OptixExceptionCodes`

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

Enumerator

OPTIX_EXCEPTION_CODE_STACK_OVERFLOW	Stack overflow of the continuation stack. no exception details.
OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED	The trace depth is exceeded. no exception details.

5.14.4.21 OptixExceptionFlags

`enum OptixExceptionFlags`

Exception flags.

See also OptixPipelineCompileOptions::exceptionFlags, OptixExceptionCodes

Enumerator

OPTIX_EXCEPTION_FLAG_NONE	No exception are enabled.
OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW	Enables exceptions check related to the continuation stack. This flag should be used when the application handles stack overflows in a user exception program as part of the normal flow of execution. For catching overflows during debugging and development, the device context validation mode should be used instead. See also OptixDeviceContextValidationMode
OPTIX_EXCEPTION_FLAG_TRACE_DEPTH	Enables exceptions check related to trace depth. This flag should be used when the application handles trace depth overflows in a user exception program as part of the normal flow of execution. For catching overflows during debugging and development, the device context validation mode should be used instead. See also OptixDeviceContextValidationMode
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( ).

5.14.4.22 OptixGeometryFlags

`enum OptixGeometryFlags`

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

Enumerator

OPTIX_GEOMETRY_FLAG_NONE	No flags set.

Enumerator

OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT	Disables the invocation of the anyhit program. Can be overridden by OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT and OPTIX_RAY_FLAG_ENFORCE_ANYHIT.
OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL	If set, an intersection with the primitive will trigger one and only one invocation of the anyhit program. Otherwise, the anyhit program may be invoked more than once.
OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING	Prevent triangles from getting culled due to their orientation. Effectively ignores ray flags OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES and OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES.

5.14.4.23 OptixHitKind

`enum OptixHitKind`

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

See also optixGetHitKind( )

Enumerator

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

5.14.4.24 OptixIndicesFormat

`enum OptixIndicesFormat`

Format of indices used 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.

5.14.4.25 OptixInstanceFlags

`enum OptixInstanceFlags`

Flags set on the OptixInstance::flags.

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

Enumerator

OPTIX_INSTANCE_FLAG_NONE	No special flag set.
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.

5.14.4.26 OptixModuleCompileState

`enum OptixModuleCompileState`

Module compilation state.

See also optixModuleGetCompilationState( ), optixModuleCreateWithTasks( )

Enumerator

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

5.14.4.27 OptixMotionFlags

`enum OptixMotionFlags`

Enum to specify motion flags.

See also OptixMotionOptions::flags.

Enumerator

OPTIX_MOTION_FLAG_NONE	
OPTIX_MOTION_FLAG_START_VANISH	
OPTIX_MOTION_FLAG_END_VANISH	

5.14.4.28 OptixOpacityMicromapArrayIndexingMode

`enum OptixOpacityMicromapArrayIndexingMode`

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

Enumerator

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

5.14.4.29 OptixOpacityMicromapFlags

`enum OptixOpacityMicromapFlags`

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

Enumerator

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

5.14.4.30 OptixOpacityMicromapFormat

`enum OptixOpacityMicromapFormat`

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

Enumerator

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

5.14.4.31 OptixPayloadSemantics

`enum OptixPayloadSemantics`

Semantic flags for a single payload word.

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

"trace_caller_write": Shaders may consume the value of the payload word passed to optixTrace by the caller. "trace_caller_read": The caller to optixTrace may read the payload word after the call to optixTrace.

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

Enumerator

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

Enumerator

OPTIX_PAYLOAD_SEMANTICS_IS_READ	
OPTIX_PAYLOAD_SEMANTICS_IS_WRITE	
OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE	

5.14.4.32 OptixPayloadTypeID

`enum OptixPayloadTypeID`

Payload type identifiers.

Enumerator

OPTIX_PAYLOAD_TYPE_DEFAULT	
OPTIX_PAYLOAD_TYPE_ID_0	
OPTIX_PAYLOAD_TYPE_ID_1	
OPTIX_PAYLOAD_TYPE_ID_2	
OPTIX_PAYLOAD_TYPE_ID_3	
OPTIX_PAYLOAD_TYPE_ID_4	
OPTIX_PAYLOAD_TYPE_ID_5	
OPTIX_PAYLOAD_TYPE_ID_6	
OPTIX_PAYLOAD_TYPE_ID_7	

5.14.4.33 OptixPixelFormat

`enum OptixPixelFormat`

Pixel formats used by the denoiser.

See also OptixImage2D::format

Enumerator

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

5.14.4.34 OptixPrimitiveType

`enum OptixPrimitiveType`

Builtin primitive types.

Enumerator

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

5.14.4.35 OptixPrimitiveTypeFlags

`enum OptixPrimitiveTypeFlags`

Builtin flags may be bitwise combined.

See also OptixPipelineCompileOptions::usesPrimitiveTypeFlags

Enumerator

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

Enumerator

OPTIX_PRIMITIVE_TYPE_FLAGS_DISPLACED_MICROMESH_TRIANGLE	Triangle with an applied displacement micromap.

5.14.4.36 OptixProgramGroupFlags

`enum OptixProgramGroupFlags`

Flags for program groups.

Enumerator

OPTIX_PROGRAM_GROUP_FLAGS_NONE	Currently there are no flags.

5.14.4.37 OptixProgramGroupKind

`enum OptixProgramGroupKind`

Distinguishes different kinds of program groups.

Enumerator

OPTIX_PROGRAM_GROUP_KIND_RAYGEN	Program group containing a raygen (RG) program. See also 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

5.14.4.38 OptixQueryFunctionTableOptions

`enum OptixQueryFunctionTableOptions`

Options that can be passed to `optixQueryFunctionTable()`

Enumerator

OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY	Placeholder (there are no options yet)

5.14.4.39 OptixRayFlags

`enum OptixRayFlags`

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

See also optixTrace( )

Enumerator

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.
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.

Enumerator

OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT	Do not intersect geometry which have an enabled anyhit program (due to not setting geometry flag OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT or setting instance flag OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT). This flag is mutually exclusive with OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT, OPTIX_RAY_FLAG_ENFORCE_ANYHIT, OPTIX_RAY_FLAG_DISABLE_ANYHIT.
OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE	Force 4-state opacity micromaps to behave as 2-state opactiy micromaps during traversal.

5.14.4.40 OptixResult

`enum OptixResult`

Result codes returned from API functions.

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

See also optixGetErrorName( ), optixGetErrorString( )

Enumerator

OPTIX_SUCCESS
OPTIX_ERROR_INVALID_VALUE
OPTIX_ERROR_HOST_OUT_OF_MEMORY
OPTIX_ERROR_INVALID_OPERATION
OPTIX_ERROR_FILE_IO_ERROR
OPTIX_ERROR_INVALID_FILE_FORMAT
OPTIX_ERROR_DISK_CACHE_INVALID_PATH
OPTIX_ERROR_DISK_CACHE_PERMISSION_ERROR
OPTIX_ERROR_DISK_CACHE_DATABASE_ERROR
OPTIX_ERROR_DISK_CACHE_INVALID_DATA
OPTIX_ERROR_LAUNCH_FAILURE
OPTIX_ERROR_INVALID_DEVICE_CONTEXT
OPTIX_ERROR_CUDA_NOT_INITIALIZED
OPTIX_ERROR_VALIDATION_FAILURE
OPTIX_ERROR_INVALID_INPUT
OPTIX_ERROR_INVALID_LAUNCH_PARAMETER
OPTIX_ERROR_INVALID_PAYLOAD_ACCESS
OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS
OPTIX_ERROR_INVALID_FUNCTION_USE

Enumerator

OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS
OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY
OPTIX_ERROR_PIPELINE_LINK_ERROR
OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE
OPTIX_ERROR_INTERNAL_COMPILER_ERROR
OPTIX_ERROR_DENOISER_MODEL_NOT_SET
OPTIX_ERROR_DENOISER_NOT_INITIALIZED
OPTIX_ERROR_NOT_COMPATIBLE
OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH
OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED
OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID
OPTIX_ERROR_NOT_SUPPORTED
OPTIX_ERROR_UNSUPPORTED_ABI_VERSION
OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH
OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS
OPTIX_ERROR_LIBRARY_NOT_FOUND
OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND
OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE
OPTIX_ERROR_DEVICE_OUT_OF_MEMORY
OPTIX_ERROR_CUDA_ERROR
OPTIX_ERROR_INTERNAL_ERROR
OPTIX_ERROR_UNKNOWN

5.14.4.41 OptixTransformFormat

`enum OptixTransformFormat`

Format of transform used in OptixBuildInputTriangleArray::transformFormat.

Enumerator

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

5.14.4.42 OptixTransformType

`enum OptixTransformType`

Transform.

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

Enumerator

OPTIX_TRANSFORM_TYPE_NONE	Not a transformation.
OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM	See also 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

5.14.4.43 OptixTraversableGraphFlags

`enum OptixTraversableGraphFlags`

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

Enumerator

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

5.14.4.44 OptixTraversableType

`enum OptixTraversableType`

Traversable Handles.

See also optixConvertPointerToTraversableHandle( )

Enumerator

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

Enumerator

OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM	SRT motion transform. See also OptixSRTMotionTransform

5.14.4.45 OptixVertexFormat

`enum OptixVertexFormat`

Format of vertices used in OptixBuildInputTriangleArray::vertexFormat.

Enumerator

OPTIX_VERTEX_FORMAT_NONE	No vertices.
OPTIX_VERTEX_FORMAT_FLOAT3	Vertices are represented by three floats.
OPTIX_VERTEX_FORMAT_FLOAT2	Vertices are represented by two floats.
OPTIX_VERTEX_FORMAT_HALF3	Vertices are represented by three halfs.
OPTIX_VERTEX_FORMAT_HALF2	Vertices are represented by two halfs.
OPTIX_VERTEX_FORMAT_SNORM16_3	
OPTIX_VERTEX_FORMAT_SNORM16_2	

6 Namespace Documentation

6.1 optix_impl Namespace Reference

Functions

- static __forceinline__ __device__ float4 optixAddFloat4 (const float4 &a, const float4 &b)
- static __forceinline__ __device__ float4 optixMulFloat4 (const float4 &a, float b)
- static __forceinline__ __device__ uint4 optixLdg (unsigned long long addr)
- template<class T >
  static __forceinline__ __device__ T optixLoadReadOnlyAlign16 (const T *ptr)
- static __forceinline__ __device__ float4 optixMultiplyRowMatrix (const float4 vec, const float4 m0, const float4 m1, const float4 m2)
- static __forceinline__ __device__ void optixGetMatrixFromSrt (float4 &m0, float4 &m1, float4 &m2, const OptixSRTData &srt)
- static __forceinline__ __device__ void optixInvertMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ void optixLoadInterpolatedMatrixKey (float4 &m0, float4 &m1, float4 &m2, const float4 *matrix, const float t1)
- static __forceinline__ __device__ void optixLoadInterpolatedSrtKey (float4 &srt0, float4 &srt1, float4 &srt2, float4 &srt3, const float4 *srt, const float t1)
- static __forceinline__ __device__ void optixResolveMotionKey (float &localt, int &key, const OptixMotionOptions &options, const float globalt)
- static __forceinline__ __device__ void optixGetInterpolatedTransformation (float4 &trf0, float4 &trf1, float4 &trf2, const OptixMatrixMotionTransform *transformData, const float time)
- static __forceinline__ __device__ void optixGetInterpolatedTransformation (float4 &trf0, float4 &trf1, float4 &trf2, const OptixSRTMotionTransform *transformData, const float time)
- static __forceinline__ __device__ void optixGetInterpolatedTransformationFromHandle (float4 &trf0, float4 &trf1, float4 &trf2, const OptixTraversableHandle handle, const float time, const bool objectToWorld)

- static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ float3 optixTransformPoint (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &p)
- static __forceinline__ __device__ float3 optixTransformVector (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &v)
- static __forceinline__ __device__ float3 optixTransformNormal (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &n)
- OPTIX_MICROMAP_INLINE_FUNC float __uint_as_float (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC unsigned int extractEvenBits (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC unsigned int prefixEor (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC void index2dbary (unsigned int index, unsigned int &u, unsigned int &v, unsigned int &w)
- OPTIX_MICROMAP_INLINE_FUNC void micro2bary (unsigned int index, unsigned int subdivisionLevel, float2 &bary0, float2 &bary1, float2 &bary2)
- OPTIX_MICROMAP_INLINE_FUNC float2 base2micro (const float2 &baseBarycentrics, const float2 microVertexBaseBarycentrics[3])

6.1.1 Function Documentation

6.1.1.1 optixAddFloat4()

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

6.1.1.2 optixGetInterpolatedTransformation() [1/2]

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

6.1.1.3 optixGetInterpolatedTransformation() [2/2]

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

6.1.1.4 optixGetInterpolatedTransformationFromHandle()

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

6.1.1.5 optixGetMatrixFromSrt()

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

6.1.1.6 optixGetObjectToWorldTransformMatrix()

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

6.1.1.7 optixGetWorldToObjectTransformMatrix()

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

6.1.1.8 optixInvertMatrix()

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

6.1.1.9 optixLdg()

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

6.1.1.10 optixLoadInterpolatedMatrixKey()

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

6.1.1.11 optixLoadInterpolatedSrtKey()

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

6.1.1.12 optixLoadReadOnlyAlign16()

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

6.1.1.13 optixMulFloat4()

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

6.1.1.14 optixMultiplyRowMatrix()

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

6.1.1.15 optixResolveMotionKey()

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

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

6.1.1.16 optixTransformNormal()

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

6.1.1.17 optixTransformPoint()

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

6.1.1.18 optixTransformVector()

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

6.2 optix_internal Namespace Reference

Classes

- struct TypePack

7 Class Documentation

7.1 OptixAabb Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.1.1 Detailed Description

AABB inputs.

7.1.2 Member Data Documentation

7.1.2.1 maxX

`float OptixAabb::maxX`

Upper extent in X direction.

7.1.2.2 maxY

`float OptixAabb::maxY`

Upper extent in Y direction.

7.1.2.3 maxZ

`float OptixAabb::maxZ`

Upper extent in Z direction.

7.1.2.4 minX

`float OptixAabb::minX`

Lower extent in X direction.

7.1.2.5 minY

`float OptixAabb::minY`

Lower extent in Y direction.

7.1.2.6 minZ

`float OptixAabb::minZ`

Lower extent in Z direction.

7.2 OptixAccelBufferSizes Struct Reference

`#include <optix_types.h>`

Public Attributes

- size_t outputSizeInBytes
- size_t tempSizeInBytes
- size_t tempUpdateSizeInBytes

7.2.1 Detailed Description

Struct for querying builder allocation requirements.

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

See also optixAccelComputeMemoryUsage( )

7.2.2 Member Data Documentation

7.2.2.1 outputSizeInBytes

`size_t OptixAccelBufferSizes::outputSizeInBytes`

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

7.2.2.2 tempSizeInBytes

```
size_t OptixAccelBufferSizes::tempSizeInBytes
```

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

7.2.2.3 tempUpdateSizeInBytes

```
size_t OptixAccelBufferSizes::tempUpdateSizeInBytes
```

The size in bytes required for the tempBuffer parameter to optixAccelBuild when doing an update (OPTIX_BUILD_OPERATION_UPDATE). This value can be different than tempSizeInBytes used for a full build. Only non-zero if OPTIX_BUILD_FLAG_ALLOW_UPDATE flag is set in OptixAccelBuildOptions.

7.3 OptixAccelBuildOptions Struct Reference

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

Public Attributes

- unsigned int buildFlags
- OptixBuildOperation operation
- OptixMotionOptions motionOptions

7.3.1 Detailed Description

Build options for acceleration structures.

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( )

7.3.2 Member Data Documentation

7.3.2.1 buildFlags

```
unsigned int OptixAccelBuildOptions::buildFlags
```

Combinations of OptixBuildFlags.

7.3.2.2 motionOptions

```
OptixMotionOptions OptixAccelBuildOptions::motionOptions
```

Options for motion.

7.3.2.3 operation

```
OptixBuildOperation OptixAccelBuildOptions::operation
```

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

7.4 OptixAccelEmitDesc Struct Reference

`#include <optix_types.h>`

Public Attributes

- CUdeviceptr result
- OptixAccelPropertyType type

7.4.1 Detailed Description

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

See also optixAccelBuild( )

7.4.2 Member Data Documentation

7.4.2.1 result

`CUdeviceptr OptixAccelEmitDesc::result`

Output buffer for the properties.

7.4.2.2 type

`OptixAccelPropertyType OptixAccelEmitDesc::type`

Requested property.

7.5 OptixBuildInput Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.5.1 Detailed Description

Build inputs.

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

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( )

7.5.2 Member Data Documentation

7.5.2.1

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

7.5.2.2 curveArray

`OptixBuildInputCurveArray OptixBuildInput::curveArray`

Curve inputs.

7.5.2.3 customPrimitiveArray

`OptixBuildInputCustomPrimitiveArray OptixBuildInput::customPrimitiveArray`

Custom primitive inputs.

7.5.2.4 instanceArray

`OptixBuildInputInstanceArray OptixBuildInput::instanceArray`

Instance and instance pointer inputs.

7.5.2.5 pad

`char OptixBuildInput::pad[1024]`

7.5.2.6 sphereArray

`OptixBuildInputSphereArray OptixBuildInput::sphereArray`

Sphere inputs.

7.5.2.7 triangleArray

`OptixBuildInputTriangleArray OptixBuildInput::triangleArray`

Triangle inputs.

7.5.2.8 type

`OptixBuildInputType OptixBuildInput::type`

The type of the build input.

7.6 OptixBuildInputCurveArray Struct Reference

`#include <optix_types.h>`

Public Attributes

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

- unsigned int endcapFlags

7.6.1 Detailed Description

Curve inputs.

A curve is a swept surface defined by a 3D spline curve and a varying width (radius). A curve (or "strand") of degree d (3=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

7.6.2 Member Data Documentation

7.6.2.1 curveType

`OptixPrimitiveType OptixBuildInputCurveArray::curveType`

Curve degree and basis.

See also OptixPrimitiveType

7.6.2.2 endcapFlags

`unsigned int OptixBuildInputCurveArray::endcapFlags`

End cap flags, see OptixCurveEndcapFlags.

7.6.2.3 flag

`unsigned int OptixBuildInputCurveArray::flag`

Combination of OptixGeometryFlags describing the primitive behavior.

7.6.2.4 indexBuffer

`CUdeviceptr OptixBuildInputCurveArray::indexBuffer`

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

7.6.2.5 indexStrideInBytes

`unsigned int OptixBuildInputCurveArray::indexStrideInBytes`

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

7.6.2.6 normalBuffers

```
const CUdeviceptr* OptixBuildInputCurveArray::normalBuffers
```

Reserved for future use.

7.6.2.7 normalStrideInBytes

```
unsigned int OptixBuildInputCurveArray::normalStrideInBytes
```

Reserved for future use.

7.6.2.8 numPrimitives

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

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

7.6.2.9 numVertices

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

Number of vertices in each buffer in vertexBuffers.

7.6.2.10 primitiveIndexOffset

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

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

7.6.2.11 vertexBuffers

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

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

7.6.2.12 vertexStrideInBytes

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

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

7.6.2.13 widthBuffers

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

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

7.6.2.14 widthStrideInBytes

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

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

7.7 OptixBuildInputCustomPrimitiveArray Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.7.1 Detailed Description

Custom primitive inputs.

See also OptixBuildInput::customPrimitiveArray

7.7.2 Member Data Documentation

7.7.2.1 aabbBuffers

`const CUdeviceptr* OptixBuildInputCustomPrimitiveArray::aabbBuffers`

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

7.7.2.2 flags

`const unsigned int* OptixBuildInputCustomPrimitiveArray::flags`

Array of flags, to specify flags per sbt record, combinations of OptixGeometryFlags describing the primitive behavior, size must match numSbtRecords.

7.7.2.3 numPrimitives

`unsigned int OptixBuildInputCustomPrimitiveArray::numPrimitives`

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

7.7.2.4 numSbtRecords

`unsigned int OptixBuildInputCustomPrimitiveArray::numSbtRecords`

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

7.7.2.5 primitiveIndexOffset

`unsigned int OptixBuildInputCustomPrimitiveArray::primitiveIndexOffset`

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

7.7.2.6 sbtIndexOffsetBuffer

`CUdeviceptr OptixBuildInputCustomPrimitiveArray::sbtIndexOffsetBuffer`

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

7.7.2.7 sbtIndexOffsetSizeInBytes

`unsigned int OptixBuildInputCustomPrimitiveArray::sbtIndexOffsetSizeInBytes`

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

7.7.2.8 sbtIndexOffsetStrideInBytes

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

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

7.7.2.9 strideInBytes

`unsigned int OptixBuildInputCustomPrimitiveArray::strideInBytes`

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

7.8 OptixBuildInputDisplacementMicromap Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.8.1 Detailed Description

Optional displacement part of a triangle array input.

7.8.2 Member Data Documentation

7.8.2.1 displacementMicromapArray

```
CUdeviceptr OptixBuildInputDisplacementMicromap::displacementMicromapArray
```

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

7.8.2.2 displacementMicromapIndexBuffer

```
CUdeviceptr OptixBuildInputDisplacementMicromap
::displacementMicromapIndexBuffer
```

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

7.8.2.3 displacementMicromapIndexOffset

```
unsigned int OptixBuildInputDisplacementMicromap
::displacementMicromapIndexOffset
```

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

7.8.2.4 displacementMicromapIndexSizeInBytes

```
unsigned int OptixBuildInputDisplacementMicromap
::displacementMicromapIndexSizeInBytes
```

2 or 4 (16 or 32 bit)

7.8.2.5 displacementMicromapIndexStrideInBytes

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

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

7.8.2.6 displacementMicromapUsageCounts

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

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

7.8.2.7 indexingMode

```
OptixDisplacementMicromapArrayIndexingMode
OptixBuildInputDisplacementMicromap::indexingMode
```

Indexing mode of triangle to displacement micromap array mapping.

7.8.2.8 numDisplacementMicromapUsageCounts

```
unsigned int OptixBuildInputDisplacementMicromap
```

`::numDisplacementMicromapUsageCounts`

Number of OptixDisplacementMicromapUsageCount entries.

7.8.2.9 triangleFlagsBuffer

`CUdeviceptr OptixBuildInputDisplacementMicromap::triangleFlagsBuffer`

Optional per-triangle flags, uint8_t per triangle, possible values defined in enum OptixDisplacementMicromapTriangleFlags.

7.8.2.10 triangleFlagsStrideInBytes

`unsigned int OptixBuildInputDisplacementMicromap::triangleFlagsStrideInBytes`

Stride in bytes for triangleFlags.

7.8.2.11 vertexBiasAndScaleBuffer

`CUdeviceptr OptixBuildInputDisplacementMicromap::vertexBiasAndScaleBuffer`

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

7.8.2.12 vertexBiasAndScaleFormat

`OptixDisplacementMicromapBiasAndScaleFormat`
`OptixBuildInputDisplacementMicromap::vertexBiasAndScaleFormat`

Format of vertex bias and direction scale.

7.8.2.13 vertexBiasAndScaleStrideInBytes

```
unsigned int OptixBuildInputDisplacementMicromap
::vertexBiasAndScaleStrideInBytes
```

Stride in bytes for vertex bias and direction scale entries.

7.8.2.14 vertexDirectionFormat

`OptixDisplacementMicromapDirectionFormat`
`OptixBuildInputDisplacementMicromap::vertexDirectionFormat`

Format of displacement vectors.

7.8.2.15 vertexDirectionsBuffer

`CUdeviceptr OptixBuildInputDisplacementMicromap::vertexDirectionsBuffer`

Per triangle-vertex displacement directions.

7.8.2.16 vertexDirectionStrideInBytes

```
unsigned int OptixBuildInputDisplacementMicromap
::vertexDirectionStrideInBytes
```

Stride between displacement vectors.

7.9 OptixBuildInputInstanceArray Struct Reference

`#include <optix_types.h>`

Public Attributes

- CUdeviceptr instances
- unsigned int numInstances
- unsigned int instanceStride

7.9.1 Detailed Description

Instance and instance pointer inputs.

See also OptixBuildInput::instanceArray

7.9.2 Member Data Documentation

7.9.2.1 instances

```
CUdeviceptr OptixBuildInputInstanceArray::instances
```

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

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

7.9.2.2 instanceStride

```
unsigned int OptixBuildInputInstanceArray::instanceStride
```

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

7.9.2.3 numInstances

```
unsigned int OptixBuildInputInstanceArray::numInstances
```

Number of elements in OptixBuildInputInstanceArray::instances.

7.10 OptixBuildInputOpacityMicromap Struct Reference

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

Public Attributes

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

7.10.1 Member Data Documentation

7.10.1.1 indexBuffer

```
CUdeviceptr OptixBuildInputOpacityMicromap::indexBuffer
```

int16 or int32 buffer specifying which opacity micromap index to use for each triangle. Instead of an actual index, one of the predefined indices OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_ (FULLY_TRANSPARENT | FULLY_OPAQUE | FULLY_UNKNOWN_TRANSPARENT | FULLY_ UNKNOWN_OPAQUE) can be used to indicate that there is no opacity micromap for this particular triangle but the triangle is in a uniform state and the selected behavior is applied to the entire triangle. This buffer is required when OptixBuildInputOpacityMicromap::indexingMode is OPTIX_OPACITY_ MICROMAP_ARRAY_INDEXING_MODE_INDEXED. Must be zero if OptixBuildInputOpacityMicromap::indexingMode is OPTIX_OPACITY_MICROMAP_ARRAY_ INDEXING_MODE_LINEAR or OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_ NONE.

7.10.1.2 indexingMode

```
OptixOpacityMicromapArrayIndexingMode OptixBuildInputOpacityMicromap
::indexingMode
```

Indexing mode of triangle to opacity micromap array mapping.

7.10.1.3 indexOffset

```
unsigned int OptixBuildInputOpacityMicromap::indexOffset
```

Constant offset to non-negative opacity micromap indices.

7.10.1.4 indexSizeInBytes

```
unsigned int OptixBuildInputOpacityMicromap::indexSizeInBytes
```

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

7.10.1.5 indexStrideInBytes

```
unsigned int OptixBuildInputOpacityMicromap::indexStrideInBytes
```

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

7.10.1.6 micromapUsageCounts

```
const OptixOpacityMicromapUsageCount* OptixBuildInputOpacityMicromap
::micromapUsageCounts
```

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

7.10.1.7 numMicromapUsageCounts

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

Number of OptixOpacityMicromapUsageCount.

7.10.1.8 opacityMicromapArray

```
CUdeviceptr OptixBuildInputOpacityMicromap::opacityMicromapArray
```

Device pointer to a opacity micromap array used by this build input array. This buffer is required when OptixBuildInputOpacityMicromap::indexingMode is OPTIX_OPACITY_MICROMAP_ARRAY_ INDEXING_MODE_LINEAR or OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_

INDEXED. Must be zero if OptixBuildInputOpacityMicromap::indexingMode is OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE.

7.11 OptixBuildInputSphereArray Struct Reference

`#include <optix_types.h>`

Public Attributes

- const CUdeviceptr * vertexBuffers
- unsigned int vertexStrideInBytes
- unsigned int numVertices
- const CUdeviceptr * radiusBuffers
- unsigned int radiusStrideInBytes
- int singleRadius
- const unsigned int * flags
- unsigned int numSbtRecords
- CUdeviceptr sbtIndexOffsetBuffer
- unsigned int sbtIndexOffsetSizeInBytes
- unsigned int sbtIndexOffsetStrideInBytes
- unsigned int primitiveIndexOffset

7.11.1 Detailed Description

Sphere inputs.

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

The vertex buffers and radius buffers point to a host array of device pointers, one per motion step. Host array size must match the number of motion keys as set in OptixMotionOptions (or an array of size 1 if OptixMotionOptions::numKeys is set to 0 or 1). Each per motion key device pointer must point to an array of vertices corresponding to the center points of the spheres, or an array of 1 or N radii. Format OPTIX_VERTEX_FORMAT_FLOAT3 is used for vertices, OPTIX_VERTEX_FORMAT_FLOAT for radii.

See also OptixBuildInput::sphereArray

7.11.2 Member Data Documentation

7.11.2.1 flags

`const unsigned int* OptixBuildInputSphereArray::flags`

Array of flags, to specify flags per sbt record, combinations of OptixGeometryFlags describing the primitive behavior, size must match numSbtRecords.

7.11.2.2 numSbtRecords

`unsigned int OptixBuildInputSphereArray::numSbtRecords`

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

7.11.2.3 numVertices

`unsigned int OptixBuildInputSphereArray::numVertices`

Number of vertices in each buffer in vertexBuffers.

7.11.2.4 primitiveIndexOffset

`unsigned int OptixBuildInputSphereArray::primitiveIndexOffset`

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

7.11.2.5 radiusBuffers

`const CUdeviceptr* OptixBuildInputSphereArray::radiusBuffers`

Parallel to vertexBuffers: a device pointer per motion step, each with numRadii float values, specifying the sphere radius corresponding to each vertex.

7.11.2.6 radiusStrideInBytes

`unsigned int OptixBuildInputSphereArray::radiusStrideInBytes`

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

7.11.2.7 sbtIndexOffsetBuffer

`CUdeviceptr OptixBuildInputSphereArray::sbtIndexOffsetBuffer`

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

7.11.2.8 sbtIndexOffsetSizeInBytes

`unsigned int OptixBuildInputSphereArray::sbtIndexOffsetSizeInBytes`

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

7.11.2.9 sbtIndexOffsetStrideInBytes

`unsigned int OptixBuildInputSphereArray::sbtIndexOffsetStrideInBytes`

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

7.11.2.10 singleRadius

`int OptixBuildInputSphereArray::singleRadius`

Boolean value indicating whether a single radius per radius buffer is used, or the number of radii in radiusBuffers equals numVertices.

7.11.2.11 vertexBuffers

`const CUdeviceptr* OptixBuildInputSphereArray::vertexBuffers`

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

7.11.2.12 vertexStrideInBytes

`unsigned int OptixBuildInputSphereArray::vertexStrideInBytes`

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

sizeof(float3).

7.12 OptixBuildInputTriangleArray Struct Reference

`#include <optix_types.h>`

Public Attributes

- const CUdeviceptr * vertexBuffers
- unsigned int numVertices
- OptixVertexFormat vertexFormat
- unsigned int vertexStrideInBytes
- CUdeviceptr indexBuffer
- unsigned int numIndexTriplets
- OptixIndicesFormat indexFormat
- unsigned int indexStrideInBytes
- CUdeviceptr preTransform
- const unsigned int * flags
- unsigned int numSbtRecords
- CUdeviceptr sbtIndexOffsetBuffer
- unsigned int sbtIndexOffsetSizeInBytes
- unsigned int sbtIndexOffsetStrideInBytes
- unsigned int primitiveIndexOffset
- OptixTransformFormat transformFormat
- OptixBuildInputOpacityMicromap opacityMicromap
- OptixBuildInputDisplacementMicromap displacementMicromap

7.12.1 Detailed Description

Triangle inputs.

See also OptixBuildInput::triangleArray

7.12.2 Member Data Documentation

7.12.2.1 displacementMicromap

`OptixBuildInputDisplacementMicromap OptixBuildInputTriangleArray ::displacementMicromap`

Optional displacement micromap inputs.

7.12.2.2 flags

`const unsigned int* OptixBuildInputTriangleArray::flags`

Array of flags, to specify flags per sbt record, combinations of OptixGeometryFlags describing the primitive behavior, size must match numSbtRecords.

7.12.2.3 indexBuffer

`CUdeviceptr OptixBuildInputTriangleArray::indexBuffer`

Optional pointer to array of 16 or 32-bit int triplets, one triplet per triangle. The minimum alignment must match the natural alignment of the type as specified in the indexFormat, i.e., for OPTIX_INDICES _FORMAT_UNSIGNED_INT3 4-byte and for OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 a 2-byte alignment.

7.12.2.4 indexFormat

`OptixIndicesFormat OptixBuildInputTriangleArray::indexFormat`

See also OptixIndicesFormat

7.12.2.5 indexStrideInBytes

`unsigned int OptixBuildInputTriangleArray::indexStrideInBytes`

Stride between triplets of indices. If set to zero, indices are assumed to be tightly packed and stride is inferred from indexFormat.

7.12.2.6 numIndexTriplets

`unsigned int OptixBuildInputTriangleArray::numIndexTriplets`

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

7.12.2.7 numSbtRecords

`unsigned int OptixBuildInputTriangleArray::numSbtRecords`

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

7.12.2.8 numVertices

`unsigned int OptixBuildInputTriangleArray::numVertices`

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

7.12.2.9 opacityMicromap

`OptixBuildInputOpacityMicromap OptixBuildInputTriangleArray ::opacityMicromap`

Optional opacity micromap inputs.

7.12.2.10 preTransform

`CUdeviceptr OptixBuildInputTriangleArray::preTransform`

Optional pointer to array of floats representing a 3x4 row major affine transformation matrix. This pointer must be a multiple of OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT.

7.12.2.11 primitiveIndexOffset

`unsigned int OptixBuildInputTriangleArray::primitiveIndexOffset`

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

7.12.2.12 sbtIndexOffsetBuffer

`CUdeviceptr OptixBuildInputTriangleArray::sbtIndexOffsetBuffer`

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

7.12.2.13 sbtIndexOffsetSizeInBytes

`unsigned int OptixBuildInputTriangleArray::sbtIndexOffsetSizeInBytes`

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

7.12.2.14 sbtIndexOffsetStrideInBytes

`unsigned int OptixBuildInputTriangleArray::sbtIndexOffsetStrideInBytes`

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

7.12.2.15 transformFormat

`OptixTransformFormat OptixBuildInputTriangleArray::transformFormat`

See also OptixTransformFormat

7.12.2.16 vertexBuffers

`const CUdeviceptr* OptixBuildInputTriangleArray::vertexBuffers`

Points to host array of device pointers, one per motion step. Host array size must match the number of motion keys as set in OptixMotionOptions (or an array of size 1 if OptixMotionOptions::numKeys is set to 0 or 1). Each per motion key device pointer must point to an array of vertices of the triangles in the format as described by vertexFormat. The minimum alignment must match the natural alignment of the type as specified in the vertexFormat, i.e., for OPTIX_VERTEX_FORMAT_FLOATX 4-byte, for all others a 2-byte alignment. However, an 16-byte stride (and buffer alignment) is recommended for vertices of format OPTIX_VERTEX_FORMAT_FLOAT3 for GAS build performance.

7.12.2.17 vertexFormat

`OptixVertexFormat OptixBuildInputTriangleArray::vertexFormat`

See also OptixVertexFormat

7.12.2.18 vertexStrideInBytes

`unsigned int OptixBuildInputTriangleArray::vertexStrideInBytes`

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

7.13 OptixBuiltinISOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixPrimitiveType builtinISModuleType
- int usesMotionBlur
- unsigned int buildFlags
- unsigned int curveEndcapFlags

7.13.1 Detailed Description

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

See also optixBuiltinISModuleGet( )

7.13.2 Member Data Documentation

7.13.2.1 buildFlags

```
unsigned int OptixBuiltinISOptions::buildFlags
```

Build flags, see OptixBuildFlags.

7.13.2.2 builtinISModuleType

```
OptixPrimitiveType OptixBuiltinISOptions::builtinISModuleType
```

7.13.2.3 curveEndcapFlags

```
unsigned int OptixBuiltinISOptions::curveEndcapFlags
```

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

7.13.2.4 usesMotionBlur

```
int OptixBuiltinISOptions::usesMotionBlur
```

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

7.14 OptixDenoiserGuideLayer Struct Reference

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

Public Attributes

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

7.14.1 Detailed Description

Guide layer for the denoiser.

See also optixDenoiserInvoke( )

7.14.2 Member Data Documentation

7.14.2.1 albedo

```
OptixImage2D OptixDenoiserGuideLayer::albedo
```

7.14.2.2 flow

```
OptixImage2D OptixDenoiserGuideLayer::flow
```

7.14.2.3 flowTrustworthiness

```
OptixImage2D OptixDenoiserGuideLayer::flowTrustworthiness
```

7.14.2.4 normal

`OptixImage2D OptixDenoiserGuideLayer::normal`

7.14.2.5 outputInternalGuideLayer

`OptixImage2D OptixDenoiserGuideLayer::outputInternalGuideLayer`

7.14.2.6 previousOutputInternalGuideLayer

`OptixImage2D OptixDenoiserGuideLayer::previousOutputInternalGuideLayer`

7.15 OptixDenoiserLayer Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixImage2D input
- OptixImage2D previousOutput
- OptixImage2D output
- OptixDenoiserAOVType type

7.15.1 Detailed Description

Input/Output layers for the denoiser.

See also optixDenoiserInvoke( )

7.15.2 Member Data Documentation

7.15.2.1 input

`OptixImage2D OptixDenoiserLayer::input`

7.15.2.2 output

`OptixImage2D OptixDenoiserLayer::output`

7.15.2.3 previousOutput

`OptixImage2D OptixDenoiserLayer::previousOutput`

7.15.2.4 type

`OptixDenoiserAOVType OptixDenoiserLayer::type`

7.16 OptixDenoiserOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int guideAlbedo
- unsigned int guideNormal
- OptixDenoiserAlphaMode denoiseAlpha

7.16.1 Detailed Description

Options used by the denoiser.

See also optixDenoiserCreate( )

7.16.2 Member Data Documentation

7.16.2.1 denoiseAlpha

`OptixDenoiserAlphaMode OptixDenoiserOptions::denoiseAlpha`

alpha denoise mode

7.16.2.2 guideAlbedo

`unsigned int OptixDenoiserOptions::guideAlbedo`

7.16.2.3 guideNormal

`unsigned int OptixDenoiserOptions::guideNormal`

7.17 OptixDenoiserParams Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.17.1 Detailed Description

Various parameters used by the denoiser.

See also optixDenoiserInvoke( )

optixDenoiserComputeIntensity( )

optixDenoiserComputeAverageColor( )

7.17.2 Member Data Documentation

7.17.2.1 blendFactor

`float OptixDenoiserParams::blendFactor`

blend factor. If set to 0 the output is 100% of the denoised input. If set to 1, the output is 100% of the unmodified input. Values between 0 and 1 will linearly interpolate between the denoised and unmodified input.

7.17.2.2 hdrAverageColor

`CUdeviceptr OptixDenoiserParams::hdrAverageColor`

this parameter is used when the OPTIX_DENOISER_MODEL_KIND_AOV model kind is set. average log color of input image, separate for RGB channels (default null pointer). points to three floats. if set to null, average log color will be calculated automatically. See hdrIntensity for tiling, this also applies

here.

7.17.2.3 hdrIntensity

`CUdeviceptr OptixDenoiserParams::hdrIntensity`

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

7.17.2.4 temporalModeUsePreviousLayers

`unsigned int OptixDenoiserParams::temporalModeUsePreviousLayers`

In temporal modes this parameter must be set to 1 if previous layers (e.g. previousOutputInternalGuideLayer) contain valid data. This is the case in the second and subsequent frames of a sequence (for example after a change of camera angle). In the first frame of such a sequence this parameter must be set to 0.

7.18 OptixDenoiserSizes Struct Reference

`#include <optix_types.h>`

Public Attributes

- size_t stateSizeInBytes
- size_t withOverlapScratchSizeInBytes
- size_t withoutOverlapScratchSizeInBytes
- unsigned int overlapWindowSizeInPixels
- size_t computeAverageColorSizeInBytes
- size_t computeIntensitySizeInBytes
- size_t internalGuideLayerPixelSizeInBytes

7.18.1 Detailed Description

Various sizes related to the denoiser.

See also optixDenoiserComputeMemoryResources( )

7.18.2 Member Data Documentation

7.18.2.1 computeAverageColorSizeInBytes

`size_t OptixDenoiserSizes::computeAverageColorSizeInBytes`

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

7.18.2.2 computeIntensitySizeInBytes

`size_t OptixDenoiserSizes::computeIntensitySizeInBytes`

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

7.18.2.3 internalGuideLayerPixelSizeInBytes

`size_t OptixDenoiserSizes::internalGuideLayerPixelSizeInBytes`

Number of bytes for each pixel in internal guide layers.

7.18.2.4 overlapWindowSizeInPixels

`unsigned int OptixDenoiserSizes::overlapWindowSizeInPixels`

Overlap on all four tile sides.

7.18.2.5 stateSizeInBytes

`size_t OptixDenoiserSizes::stateSizeInBytes`

Size of state memory passed to optixDenoiserSetup, optixDenoiserInvoke.

7.18.2.6 withoutOverlapScratchSizeInBytes

`size_t OptixDenoiserSizes::withoutOverlapScratchSizeInBytes`

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

7.18.2.7 withOverlapScratchSizeInBytes

`size_t OptixDenoiserSizes::withOverlapScratchSizeInBytes`

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

7.19 OptixDeviceContextOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.19.1 Detailed Description

Parameters used for optixDeviceContextCreate( )

See also optixDeviceContextCreate( )

7.19.2 Member Data Documentation

7.19.2.1 logCallbackData

`void* OptixDeviceContextOptions::logCallbackData`

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

7.19.2.2 logCallbackFunction

`OptixLogCallback OptixDeviceContextOptions::logCallbackFunction`

Function pointer used when OptiX wishes to generate messages.

7.19.2.3 logCallbackLevel

```
int OptixDeviceContextOptions::logCallbackLevel
```

Maximum callback level to generate message for (see OptixLogCallback)

7.19.2.4 validationMode

```
OptixDeviceContextValidationMode OptixDeviceContextOptions::validationMode
```

Validation mode of context.

7.20 OptixDisplacementMicromapArrayBuildInput Struct Reference

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

Public Attributes

- OptixDisplacementMicromapFlags flags
- CUdeviceptr displacementValuesBuffer
- CUdeviceptr perDisplacementMicromapDescBuffer
- unsigned int perDisplacementMicromapDescStrideInBytes
- unsigned int numDisplacementMicromapHistogramEntries
- const OptixDisplacementMicromapHistogramEntry * displacementMicromapHistogramEntries

7.20.1 Detailed Description

Inputs to displacement micromaps array construction.

7.20.2 Member Data Documentation

7.20.2.1 displacementMicromapHistogramEntries

```
const OptixDisplacementMicromapHistogramEntry*
OptixDisplacementMicromapArrayBuildInput
::displacementMicromapHistogramEntries
```

Histogram over DMMs for input format and subdivision combinations. Counts of histogram bins with equal format and subdivision combinations are added together.

7.20.2.2 displacementValuesBuffer

```
CUdeviceptr OptixDisplacementMicromapArrayBuildInput
::displacementValuesBuffer
```

128 byte aligned pointer for displacement micromap raw input data.

7.20.2.3 flags

```
OptixDisplacementMicromapFlags OptixDisplacementMicromapArrayBuildInput
::flags
```

Flags that apply to all displacement micromaps in array.

7.20.2.4 numDisplacementMicromapHistogramEntries

```
unsigned int OptixDisplacementMicromapArrayBuildInput
::numDisplacementMicromapHistogramEntries
```

Number of OptixDisplacementMicromapHistogramEntry entries.

7.20.2.5 perDisplacementMicromapDescBuffer

```
CUdeviceptr OptixDisplacementMicromapArrayBuildInput
::perDisplacementMicromapDescBuffer
```

Descriptors for interpreting raw input data, one OptixDisplacementMicromapDesc entry required per displacement micromap. This device pointer must be a multiple of OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT.

7.20.2.6 perDisplacementMicromapDescStrideInBytes

```
unsigned int OptixDisplacementMicromapArrayBuildInput
::perDisplacementMicromapDescStrideInBytes
```

Stride between OptixDisplacementMicromapDesc in perDisplacementMicromapDescBuffer If set to zero, the displacement micromap descriptors are assumed to be tightly packed and the stride is assumed to be sizeof(OptixDisplacementMicromapDesc). This stride must be a multiple of OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT.

7.21 OptixDisplacementMicromapDesc Struct Reference

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

Public Attributes

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

7.21.1 Member Data Documentation

7.21.1.1 byteOffset

```
unsigned int OptixDisplacementMicromapDesc::byteOffset
```

Block is located at displacementValuesBuffer + byteOffset.

7.21.1.2 format

```
unsigned short OptixDisplacementMicromapDesc::format
```

Format (OptixDisplacementMicromapFormat)

7.21.1.3 subdivisionLevel

```
unsigned short OptixDisplacementMicromapDesc::subdivisionLevel
```

Number of micro-triangles is 4^level. Valid levels are [0, 5].

7.22 OptixDisplacementMicromapHistogramEntry Struct Reference

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

Public Attributes

- unsigned int count
- unsigned int subdivisionLevel
- OptixDisplacementMicromapFormat format

7.22.1 Detailed Description

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

7.22.2 Member Data Documentation

7.22.2.1 count

```
unsigned int OptixDisplacementMicromapHistogramEntry::count
```

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

7.22.2.2 format

```
OptixDisplacementMicromapFormat OptixDisplacementMicromapHistogramEntry
::format
```

Displacement micromap format.

7.22.2.3 subdivisionLevel

```
unsigned int OptixDisplacementMicromapHistogramEntry::subdivisionLevel
```

Number of micro-triangles is 4^level. Valid levels are [0, 5].

7.23 OptixDisplacementMicromapUsageCount Struct Reference

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

Public Attributes

- unsigned int count
- unsigned int subdivisionLevel
- OptixDisplacementMicromapFormat format

7.23.1 Detailed Description

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

7.23.2 Member Data Documentation

7.23.2.1 count

```
unsigned int OptixDisplacementMicromapUsageCount::count
```

Number of displacement micromaps with this format and subdivision level referenced by triangles in the corresponding triangle build input at AS build time.

7.23.2.2 format

```
OptixDisplacementMicromapFormat OptixDisplacementMicromapUsageCount::format
```

Displacement micromaps format.

7.23.2.3 subdivisionLevel

`unsigned int OptixDisplacementMicromapUsageCount::subdivisionLevel`

Number of micro-triangles is 4^level. Valid levels are [0, 5].

7.24 OptixFunctionTable Struct Reference

`#include <optix_function_table.h>`

Public Attributes

Error handling

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

Device context

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

Modules

- OptixResult(* optixModuleCreate)(OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module)
- OptixResult(* optixModuleCreateWithTasks)(OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module, OptixTask *firstTask)
- OptixResult(* optixModuleGetCompilationState)(OptixModule module, OptixModuleCompileState *state)
- OptixResult(* optixModuleDestroy)(OptixModule module)
- OptixResult(* optixBuiltinISModuleGet)(OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixBuiltinISOptions *builtinISOptions, OptixModule *builtinModule)

Tasks

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

Program groups

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

Pipeline

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

Acceleration structures

- OptixResult(∗ optixAccelComputeMemoryUsage)(OptixDeviceContext context, const OptixAccelBuildOptions ∗accelOptions, const OptixBuildInput ∗buildInputs, unsigned int numBuildInputs, OptixAccelBufferSizes ∗bufferSizes)
- OptixResult(∗ optixAccelBuild)(OptixDeviceContext context, CUstream stream, const OptixAccelBuildOptions ∗accelOptions, const OptixBuildInput ∗buildInputs, unsigned int numBuildInputs, CUdeviceptr tempBuffer, size_t tempBufferSizeInBytes, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle ∗outputHandle, const OptixAccelEmitDesc ∗emittedProperties, unsigned int numEmittedProperties)
- OptixResult(∗ optixAccelGetRelocationInfo)(OptixDeviceContext context, OptixTraversableHandle handle, OptixRelocationInfo ∗info)
- OptixResult(∗ optixCheckRelocationCompatibility)(OptixDeviceContext context, const OptixRelocationInfo ∗info, int ∗compatible)
- OptixResult(∗ optixAccelRelocate)(OptixDeviceContext context, CUstream stream, const OptixRelocationInfo ∗info, const OptixRelocateInput ∗relocateInputs, size_t numRelocateInputs, CUdeviceptr targetAccel, size_t targetAccelSizeInBytes, OptixTraversableHandle ∗targetHandle)
- OptixResult(∗ optixAccelCompact)(OptixDeviceContext context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle ∗outputHandle)
- OptixResult(∗ optixAccelEmitProperty)(OptixDeviceContext context, CUstream stream, OptixTraversableHandle handle, const OptixAccelEmitDesc ∗emittedProperty)
- OptixResult(∗ optixConvertPointerToTraversableHandle)(OptixDeviceContext onDevice, CUdeviceptr pointer, OptixTraversableType traversableType, OptixTraversableHandle ∗traversableHandle)
- OptixResult(∗ optixOpacityMicromapArrayComputeMemoryUsage)(OptixDeviceContext context, const OptixOpacityMicromapArrayBuildInput ∗buildInput, OptixMicromapBufferSizes ∗bufferSizes)
- OptixResult(∗ optixOpacityMicromapArrayBuild)(OptixDeviceContext context, CUstream stream, const OptixOpacityMicromapArrayBuildInput ∗buildInput, const OptixMicromapBuffers ∗buffers)

- OptixResult(∗ optixOpacityMicromapArrayGetRelocationInfo)(OptixDeviceContext context, CUdeviceptr opacityMicromapArray, OptixRelocationInfo ∗info)
- OptixResult(∗ optixOpacityMicromapArrayRelocate)(OptixDeviceContext context, CUstream stream, const OptixRelocationInfo ∗info, CUdeviceptr targetOpacityMicromapArray, size_t targetOpacityMicromapArraySizeInBytes)
- OptixResult(∗ optixDisplacementMicromapArrayComputeMemoryUsage)(OptixDeviceContext context, const OptixDisplacementMicromapArrayBuildInput ∗buildInput, OptixMicromapBufferSizes ∗bufferSizes)
- OptixResult(∗ optixDisplacementMicromapArrayBuild)(OptixDeviceContext context, CUstream stream, const OptixDisplacementMicromapArrayBuildInput ∗buildInput, const OptixMicromapBuffers ∗buffers)

Launch

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

Denoiser

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

7.24.1 Detailed Description

The function table containing all API functions.

See optixInit( ) and optixInitWithHandle( ).

7.24.2 Member Data Documentation

7.24.2.1 optixAccelBuild

```
OptixResult(∗ OptixFunctionTable::optixAccelBuild) (OptixDeviceContext
context, CUstream stream, const OptixAccelBuildOptions ∗accelOptions, const
```

```
OptixBuildInput *buildInputs, unsigned int numBuildInputs, CUdeviceptr
tempBuffer, size_t tempBufferSizeInBytes, CUdeviceptr outputBuffer, size_t
outputBufferSizeInBytes, OptixTraversableHandle *outputHandle, const
OptixAccelEmitDesc *emittedProperties, unsigned int numEmittedProperties)
```

See optixAccelBuild( ).

7.24.2.2 optixAccelCompact

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

See optixAccelCompact( ).

7.24.2.3 optixAccelComputeMemoryUsage

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

See optixAccelComputeMemoryUsage( ).

7.24.2.4 optixAccelEmitProperty

```
OptixResult(* OptixFunctionTable::optixAccelEmitProperty)
(OptixDeviceContext context, CUstream stream, OptixTraversableHandle handle,
const OptixAccelEmitDesc *emittedProperty)
```

See optixAccelComputeMemoryUsage( ).

7.24.2.5 optixAccelGetRelocationInfo

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

See optixAccelGetRelocationInfo( ).

7.24.2.6 optixAccelRelocate

```
OptixResult(* OptixFunctionTable::optixAccelRelocate) (OptixDeviceContext
context, CUstream stream, const OptixRelocationInfo *info, const
OptixRelocateInput *relocateInputs, size_t numRelocateInputs, CUdeviceptr
targetAccel, size_t targetAccelSizeInBytes, OptixTraversableHandle
*targetHandle)
```

See optixAccelRelocate( ).

7.24.2.7 optixBuiltinISModuleGet

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

See optixBuiltinISModuleGet( ).

7.24.2.8 optixCheckRelocationCompatibility

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

See optixCheckRelocationCompatibility( ).

7.24.2.9 optixConvertPointerToTraversableHandle

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

See optixConvertPointerToTraversableHandle( ).

7.24.2.10 optixDenoiserComputeAverageColor

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

See optixDenoiserComputeAverageColor( ).

7.24.2.11 optixDenoiserComputeIntensity

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

See optixDenoiserComputeIntensity( ).

7.24.2.12 optixDenoiserComputeMemoryResources

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

See optixDenoiserComputeMemoryResources( ).

7.24.2.13 optixDenoiserCreate

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

See optixDenoiserCreate( ).

7.24.2.14 optixDenoiserCreateWithUserModel

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

See optixDenoiserCreateWithUserModel( ).

7.24.2.15 optixDenoiserDestroy

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

See optixDenoiserDestroy( ).

7.24.2.16 optixDenoiserInvoke

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

See optixDenoiserInvoke( ).

7.24.2.17 optixDenoiserSetup

```
OptixResult(* OptixFunctionTable::optixDenoiserSetup) (OptixDenoiser
denoiser, CUstream stream, unsigned int inputWidth, unsigned int
inputHeight, CUdeviceptr state, size_t stateSizeInBytes, CUdeviceptr
scratch, size_t scratchSizeInBytes)
```

See optixDenoiserSetup( ).

7.24.2.18 optixDeviceContextCreate

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

See optixDeviceContextCreate( ).

7.24.2.19 optixDeviceContextDestroy

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

See optixDeviceContextDestroy( ).

7.24.2.20 optixDeviceContextGetCacheDatabaseSizes

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

See optixDeviceContextGetCacheDatabaseSizes( ).

7.24.2.21 optixDeviceContextGetCacheEnabled

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

See optixDeviceContextGetCacheEnabled( ).

7.24.2.22 optixDeviceContextGetCacheLocation

```
OptixResult(* OptixFunctionTable::optixDeviceContextGetCacheLocation)
```

```
(OptixDeviceContext context, char *location, size_t locationSize)
```

See optixDeviceContextGetCacheLocation( ).

7.24.2.23 optixDeviceContextGetProperty

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

See optixDeviceContextGetProperty( ).

7.24.2.24 optixDeviceContextSetCacheDatabaseSizes

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

See optixDeviceContextSetCacheDatabaseSizes( ).

7.24.2.25 optixDeviceContextSetCacheEnabled

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

See optixDeviceContextSetCacheEnabled( ).

7.24.2.26 optixDeviceContextSetCacheLocation

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

See optixDeviceContextSetCacheLocation( ).

7.24.2.27 optixDeviceContextSetLogCallback

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

See optixDeviceContextSetLogCallback( ).

7.24.2.28 optixDisplacementMicromapArrayBuild

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

See optixDisplacementMicromapArrayBuild( ).

7.24.2.29 optixDisplacementMicromapArrayComputeMemoryUsage

```
OptixResult(* OptixFunctionTable
::optixDisplacementMicromapArrayComputeMemoryUsage) (OptixDeviceContext
context, const OptixDisplacementMicromapArrayBuildInput *buildInput,
OptixMicromapBufferSizes *bufferSizes)
```

See optixDisplacementMicromapArrayComputeMemoryUsage( ).

7.24.2.30 optixGetErrorName

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

See optixGetErrorName( ).

7.24.2.31 optixGetErrorString

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

See optixGetErrorString( ).

7.24.2.32 optixLaunch

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

See optixConvertPointerToTraversableHandle( ).

7.24.2.33 optixModuleCreate

```
OptixResult(* OptixFunctionTable::optixModuleCreate) (OptixDeviceContext
context, const OptixModuleCompileOptions *moduleCompileOptions, const
OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size
_t inputSize, char *logString, size_t *logStringSize, OptixModule *module)
```

See optixModuleCreate( ).

7.24.2.34 optixModuleCreateWithTasks

```
OptixResult(* OptixFunctionTable::optixModuleCreateWithTasks)
(OptixDeviceContext context, const OptixModuleCompileOptions
*moduleCompileOptions, const OptixPipelineCompileOptions
*pipelineCompileOptions, const char *input, size_t inputSize, char
*logString, size_t *logStringSize, OptixModule *module, OptixTask
*firstTask)
```

See optixModuleCreateWithTasks( ).

7.24.2.35 optixModuleDestroy

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

See optixModuleDestroy( ).

7.24.2.36 optixModuleGetCompilationState

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

See optixModuleGetCompilationState( ).

7.24.2.37 optixOpacityMicromapArrayBuild

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

```
*buffers)
```

See optixOpacityMicromapArrayBuild( ).

7.24.2.38 optixOpacityMicromapArrayComputeMemoryUsage

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

See optixOpacityMicromapArrayComputeMemoryUsage( ).

7.24.2.39 optixOpacityMicromapArrayGetRelocationInfo

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

See optixOpacityMicromapArrayGetRelocationInfo( ).

7.24.2.40 optixOpacityMicromapArrayRelocate

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

See optixOpacityMicromapArrayRelocate( ).

7.24.2.41 optixPipelineCreate

```
OptixResult(* OptixFunctionTable::optixPipelineCreate) (OptixDeviceContext
context, const OptixPipelineCompileOptions *pipelineCompileOptions, const
OptixPipelineLinkOptions *pipelineLinkOptions, const OptixProgramGroup
*programGroups, unsigned int numProgramGroups, char *logString, size_t
*logStringSize, OptixPipeline *pipeline)
```

See optixPipelineCreate( ).

7.24.2.42 optixPipelineDestroy

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

See optixPipelineDestroy( ).

7.24.2.43 optixPipelineSetStackSize

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

See optixPipelineSetStackSize( ).

7.24.2.44 optixProgramGroupCreate

```
OptixResult(* OptixFunctionTable::optixProgramGroupCreate)
```

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

See optixProgramGroupCreate( ).

7.24.2.45 optixProgramGroupDestroy

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

See optixProgramGroupDestroy( ).

7.24.2.46 optixProgramGroupGetStackSize

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

See optixProgramGroupGetStackSize( ).

7.24.2.47 optixSbtRecordPackHeader

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

See optixConvertPointerToTraversableHandle( ).

7.24.2.48 optixTaskExecute

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

See optixTaskExecute( ).

7.25 OptixImage2D Struct Reference

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

Public Attributes

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

7.25.1 Detailed Description

Image descriptor used by the denoiser.

See also optixDenoiserInvoke( ), optixDenoiserComputeIntensity( )

7.25.2 Member Data Documentation

7.25.2.1 data

`CUdeviceptr OptixImage2D::data`

Pointer to the actual pixel data.

7.25.2.2 format

`OptixPixelFormat OptixImage2D::format`

Pixel format.

7.25.2.3 height

`unsigned int OptixImage2D::height`

Height of the image (in pixels)

7.25.2.4 pixelStrideInBytes

`unsigned int OptixImage2D::pixelStrideInBytes`

Stride between subsequent pixels of the image (in bytes). If set to 0, dense packing (no gaps) is assumed. For pixel format OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER it must be set to OptixDenoiserSizes::internalGuideLayerPixelSizeInBytes.

7.25.2.5 rowStrideInBytes

`unsigned int OptixImage2D::rowStrideInBytes`

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

7.25.2.6 width

`unsigned int OptixImage2D::width`

Width of the image (in pixels)

7.26 OptixInstance Struct Reference

`#include <optix_types.h>`

Public Attributes

- float transform [12]
- unsigned int instanceId
- unsigned int sbtOffset
- unsigned int visibilityMask
- unsigned int flags
- OptixTraversableHandle traversableHandle
- unsigned int pad [2]

7.26.1 Detailed Description

Instances.

See also OptixBuildInputInstanceArray::instances

7.26.2 Member Data Documentation

7.26.2.1 flags

`unsigned int OptixInstance::flags`

Any combination of OptixInstanceFlags is allowed.

7.26.2.2 instanceId

`unsigned int OptixInstance::instanceId`

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

7.26.2.3 pad

`unsigned int OptixInstance::pad[2]`

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

7.26.2.4 sbtOffset

`unsigned int OptixInstance::sbtOffset`

SBT record offset. In a traversable graph with multiple levels of instance acceleration structure (IAS) objects, offsets are summed together. The maximal SBT offset can be queried using OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET.

7.26.2.5 transform

`float OptixInstance::transform[12]`

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

7.26.2.6 traversableHandle

`OptixTraversableHandle OptixInstance::traversableHandle`

Set with an OptixTraversableHandle.

7.26.2.7 visibilityMask

`unsigned int OptixInstance::visibilityMask`

Visibility mask. If rayMask & instanceMask == 0 the instance is culled. The number of available bits can be queried using OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK.

7.27 OptixMatrixMotionTransform Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.27.1 Detailed Description

Represents a matrix motion transformation.

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

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

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

See also optixConvertPointerToTraversableHandle( )

7.27.2 Member Data Documentation

7.27.2.1 child

`OptixTraversableHandle OptixMatrixMotionTransform::child`

The traversable that is transformed by this transformation.

7.27.2.2 motionOptions

`OptixMotionOptions OptixMatrixMotionTransform::motionOptions`

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

7.27.2.3 pad

`unsigned int OptixMatrixMotionTransform::pad[3]`

Padding to make the transformation 16 byte aligned.

7.27.2.4 transform

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

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

7.28 OptixMicromapBuffers Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.28.1 Detailed Description

Buffer inputs for opacity/displacement micromap array builds.

7.28.2 Member Data Documentation

7.28.2.1 output

```
CUdeviceptr OptixMicromapBuffers::output
```

Output buffer.

7.28.2.2 outputSizeInBytes

```
size_t OptixMicromapBuffers::outputSizeInBytes
```

Output buffer size.

7.28.2.3 temp

```
CUdeviceptr OptixMicromapBuffers::temp
```

Temp buffer.

7.28.2.4 tempSizeInBytes

```
size_t OptixMicromapBuffers::tempSizeInBytes
```

Temp buffer size.

7.29 OptixMicromapBufferSizes Struct Reference

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

Public Attributes

- size_t outputSizeInBytes
- size_t tempSizeInBytes

7.29.1 Detailed Description

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

7.29.2 Member Data Documentation

7.29.2.1 outputSizeInBytes

```
size_t OptixMicromapBufferSizes::outputSizeInBytes
```

7.29.2.2 tempSizeInBytes

```
size_t OptixMicromapBufferSizes::tempSizeInBytes
```

7.30 OptixModuleCompileBoundValueEntry Struct Reference

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

Public Attributes

- size_t pipelineParamOffsetInBytes

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

7.30.1 Detailed Description

Struct for specifying specializations for pipelineParams as specified in OptixPipelineCompileOptions ::pipelineLaunchParamsVariableName.

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

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

If more than one bound value overlaps or the size of a bound value is equal to 0, an OPTIX_ERROR_ INVALID_VALUE will be returned from optixModuleCreate.

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

See also OptixModuleCompileOptions

7.30.2 Member Data Documentation

7.30.2.1 annotation

```
const char* OptixModuleCompileBoundValueEntry::annotation
```

7.30.2.2 boundValuePtr

```
const void* OptixModuleCompileBoundValueEntry::boundValuePtr
```

7.30.2.3 pipelineParamOffsetInBytes

```
size_t OptixModuleCompileBoundValueEntry::pipelineParamOffsetInBytes
```

7.30.2.4 sizeInBytes

```
size_t OptixModuleCompileBoundValueEntry::sizeInBytes
```

7.31 OptixModuleCompileOptions Struct Reference

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

Public Attributes

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

7.31.1 Detailed Description

Compilation options for module.

See also optixModuleCreate( )

7.31.2 Member Data Documentation

7.31.2.1 boundValues

```
const OptixModuleCompileBoundValueEntry* OptixModuleCompileOptions
::boundValues
```

Ingored if numBoundValues is set to 0.

7.31.2.2 debugLevel

```
OptixCompileDebugLevel OptixModuleCompileOptions::debugLevel
```

Generate debug information.

7.31.2.3 maxRegisterCount

```
int OptixModuleCompileOptions::maxRegisterCount
```

Maximum number of registers allowed when compiling to SASS. Set to 0 for no explicit limit. May vary within a pipeline.

7.31.2.4 numBoundValues

```
unsigned int OptixModuleCompileOptions::numBoundValues
```

set to 0 if unused

7.31.2.5 numPayloadTypes

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

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

7.31.2.6 optLevel

```
OptixCompileOptimizationLevel OptixModuleCompileOptions::optLevel
```

Optimization level. May vary within a pipeline.

7.31.2.7 payloadTypes

```
const OptixPayloadType* OptixModuleCompileOptions::payloadTypes
```

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

7.32 OptixMotionOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.32.1 Detailed Description

Motion options.

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

7.32.2 Member Data Documentation

7.32.2.1 flags

`unsigned short OptixMotionOptions::flags`

Combinations of OptixMotionFlags.

7.32.2.2 numKeys

`unsigned short OptixMotionOptions::numKeys`

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

7.32.2.3 timeBegin

`float OptixMotionOptions::timeBegin`

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

7.32.2.4 timeEnd

`float OptixMotionOptions::timeEnd`

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

7.33 OptixOpacityMicromapArrayBuildInput Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.33.1 Detailed Description

Inputs to opacity micromap array construction.

7.33.2 Member Data Documentation

7.33.2.1 flags

```
unsigned int OptixOpacityMicromapArrayBuildInput::flags
```

Applies to all opacity micromaps in array.

7.33.2.2 inputBuffer

```
CUdeviceptr OptixOpacityMicromapArrayBuildInput::inputBuffer
```

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

7.33.2.3 micromapHistogramEntries

```
const OptixOpacityMicromapHistogramEntry*
OptixOpacityMicromapArrayBuildInput::micromapHistogramEntries
```

Histogram over opacity micromaps of input format and subdivision combinations. Counts of entries with equal format and subdivision combination (duplicates) are added together.

7.33.2.4 numMicromapHistogramEntries

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

Number of OptixOpacityMicromapHistogramEntry.

7.33.2.5 perMicromapDescBuffer

```
CUdeviceptr OptixOpacityMicromapArrayBuildInput::perMicromapDescBuffer
```

One OptixOpacityMicromapDesc entry per opacity micromap. This device pointer must be a multiple of OPTIX_OPACITY_MICROMAP_DESC_BYTE_ALIGNMENT.

7.33.2.6 perMicromapDescStrideInBytes

```
unsigned int OptixOpacityMicromapArrayBuildInput
::perMicromapDescStrideInBytes
```

Stride between OptixOpacityMicromapDescs in perOmDescBuffer. If set to zero, the opacity micromap descriptors are assumed to be tightly packed and the stride is assumed to be sizeof(OptixOpacityMicromapDesc). This stride must be a multiple of OPTIX_OPACITY_MICROMAP_DESC_BYTE_ALIGNMENT.

7.34 OptixOpacityMicromapDesc Struct Reference

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

Public Attributes

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

7.34.1 Detailed Description

Opacity micromap descriptor.

7.34.2 Member Data Documentation

7.34.2.1 byteOffset

`unsigned int OptixOpacityMicromapDesc::byteOffset`

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

7.34.2.2 format

`unsigned short OptixOpacityMicromapDesc::format`

OptixOpacityMicromapFormat.

7.34.2.3 subdivisionLevel

`unsigned short OptixOpacityMicromapDesc::subdivisionLevel`

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

7.35 OptixOpacityMicromapHistogramEntry Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int count
- unsigned int subdivisionLevel
- OptixOpacityMicromapFormat format

7.35.1 Detailed Description

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

7.35.2 Member Data Documentation

7.35.2.1 count

`unsigned int OptixOpacityMicromapHistogramEntry::count`

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

7.35.2.2 format

`OptixOpacityMicromapFormat OptixOpacityMicromapHistogramEntry::format`

Opacity micromap format.

7.35.2.3 subdivisionLevel

`unsigned int OptixOpacityMicromapHistogramEntry::subdivisionLevel`

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

7.36 OptixOpacityMicromapUsageCount Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int count
- unsigned int subdivisionLevel
- OptixOpacityMicromapFormat format

7.36.1 Detailed Description

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

7.36.2 Member Data Documentation

7.36.2.1 count

`unsigned int OptixOpacityMicromapUsageCount::count`

Number of opacity micromaps with this format and subdivision level referenced by triangles in the corresponding triangle build input at AS build time.

7.36.2.2 format

`OptixOpacityMicromapFormat OptixOpacityMicromapUsageCount::format`

opacity micromap format.

7.36.2.3 subdivisionLevel

`unsigned int OptixOpacityMicromapUsageCount::subdivisionLevel`

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

7.37 OptixPayloadType Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int numPayloadValues
- const unsigned int * payloadSemantics

7.37.1 Detailed Description

Specifies a single payload type.

7.37.2 Member Data Documentation

7.37.2.1 numPayloadValues

`unsigned int OptixPayloadType::numPayloadValues`

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

7.37.2.2 payloadSemantics

`const unsigned int* OptixPayloadType::payloadSemantics`

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

7.38 OptixPipelineCompileOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.38.1 Detailed Description

Compilation options for all modules of a pipeline.

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

See also optixModuleCreate( ), optixPipelineCreate( )

7.38.2 Member Data Documentation

7.38.2.1 allowOpacityMicromaps

`int OptixPipelineCompileOptions::allowOpacityMicromaps`

Boolean value indicating whether opacity micromaps could be used.

7.38.2.2 exceptionFlags

`unsigned int OptixPipelineCompileOptions::exceptionFlags`

A bitmask of OptixExceptionFlags indicating which exceptions are enabled.

7.38.2.3 numAttributeValues

`int OptixPipelineCompileOptions::numAttributeValues`

How much storage, in 32b words, to make available for the attributes. The minimum number is 2. Values below that will automatically be changed to 2. [2..8].

7.38.2.4 numPayloadValues

`int OptixPipelineCompileOptions::numPayloadValues`

How much storage, in 32b words, to make available for the payload, [0..32] Must be zero if numPayloadTypes is not zero.

7.38.2.5 pipelineLaunchParamsVariableName

`const char* OptixPipelineCompileOptions::pipelineLaunchParamsVariableName`

The name of the pipeline parameter variable. If 0, no pipeline parameter will be available. This will be ignored if the launch param variable was optimized out or was not found in the modules linked to the pipeline.

7.38.2.6 traversableGraphFlags

`unsigned int OptixPipelineCompileOptions::traversableGraphFlags`

Traversable graph bitfield. See OptixTraversableGraphFlags.

7.38.2.7 usesMotionBlur

`int OptixPipelineCompileOptions::usesMotionBlur`

Boolean value indicating whether motion blur could be used.

7.38.2.8 usesPrimitiveTypeFlags

`unsigned int OptixPipelineCompileOptions::usesPrimitiveTypeFlags`

Bit field enabling primitive types. See OptixPrimitiveTypeFlags. Setting to zero corresponds to enabling OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM and OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE.

7.39 OptixPipelineLinkOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int maxTraceDepth

7.39.1 Detailed Description

Link options for a pipeline.

See also optixPipelineCreate( )

7.39.2 Member Data Documentation

7.39.2.1 maxTraceDepth

`unsigned int OptixPipelineLinkOptions::maxTraceDepth`

Maximum trace recursion depth. 0 means a ray generation program can be launched, but can't trace any rays. The maximum allowed value is 31.

7.40 OptixProgramGroupCallables Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixModule moduleDC
- const char * entryFunctionNameDC
- OptixModule moduleCC

- const char * entryFunctionNameCC

7.40.1 Detailed Description

Program group representing callables.

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

See also #OptixProgramGroupDesc::callables

7.40.2 Member Data Documentation

7.40.2.1 entryFunctionNameCC

```
const char* OptixProgramGroupCallables::entryFunctionNameCC
```

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

7.40.2.2 entryFunctionNameDC

```
const char* OptixProgramGroupCallables::entryFunctionNameDC
```

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

7.40.2.3 moduleCC

```
OptixModule OptixProgramGroupCallables::moduleCC
```

Module holding the continuation callable (CC) program.

7.40.2.4 moduleDC

```
OptixModule OptixProgramGroupCallables::moduleDC
```

Module holding the direct callable (DC) program.

7.41 OptixProgramGroupDesc Struct Reference

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

Public Attributes

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

7.41.1 Detailed Description

Descriptor for program groups.

7.41.2 Member Data Documentation

7.41.2.1

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

7.41.2.2 callables

`OptixProgramGroupCallables OptixProgramGroupDesc::callables`

See also OPTIX_PROGRAM_GROUP_KIND_CALLABLES

7.41.2.3 exception

`OptixProgramGroupSingleModule OptixProgramGroupDesc::exception`

See also OPTIX_PROGRAM_GROUP_KIND_EXCEPTION

7.41.2.4 flags

`unsigned int OptixProgramGroupDesc::flags`

See OptixProgramGroupFlags.

7.41.2.5 hitgroup

`OptixProgramGroupHitgroup OptixProgramGroupDesc::hitgroup`

See also OPTIX_PROGRAM_GROUP_KIND_HITGROUP

7.41.2.6 kind

`OptixProgramGroupKind OptixProgramGroupDesc::kind`

The kind of program group.

7.41.2.7 miss

`OptixProgramGroupSingleModule OptixProgramGroupDesc::miss`

See also OPTIX_PROGRAM_GROUP_KIND_MISS

7.41.2.8 raygen

`OptixProgramGroupSingleModule OptixProgramGroupDesc::raygen`

See also OPTIX_PROGRAM_GROUP_KIND_RAYGEN

7.42 OptixProgramGroupHitgroup Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.42.1 Detailed Description

Program group representing the hitgroup.

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

See also OptixProgramGroupDesc::hitgroup

7.42.2 Member Data Documentation

7.42.2.1 entryFunctionNameAH

`const char* OptixProgramGroupHitgroup::entryFunctionNameAH`

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

7.42.2.2 entryFunctionNameCH

`const char* OptixProgramGroupHitgroup::entryFunctionNameCH`

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

7.42.2.3 entryFunctionNameIS

`const char* OptixProgramGroupHitgroup::entryFunctionNameIS`

Entry function name of the intersection (IS) program.

7.42.2.4 moduleAH

`OptixModule OptixProgramGroupHitgroup::moduleAH`

Module holding the any hit (AH) program.

7.42.2.5 moduleCH

`OptixModule OptixProgramGroupHitgroup::moduleCH`

Module holding the closest hit (CH) program.

7.42.2.6 moduleIS

`OptixModule OptixProgramGroupHitgroup::moduleIS`

Module holding the intersection (Is) program.

7.43 OptixProgramGroupOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

- const OptixPayloadType * payloadType

7.43.1 Detailed Description

Program group options.

See also optixProgramGroupCreate( )

7.43.2 Member Data Documentation

7.43.2.1 payloadType

`const OptixPayloadType* OptixProgramGroupOptions::payloadType`

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

See also optixSetPayloadTypes or otherwise all types specified in

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

7.44 OptixProgramGroupSingleModule Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixModule module
- const char * entryFunctionName

7.44.1 Detailed Description

Program group representing a single module.

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

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

7.44.2 Member Data Documentation

7.44.2.1 entryFunctionName

`const char* OptixProgramGroupSingleModule::entryFunctionName`

Entry function name of the single program.

7.44.2.2 module

`OptixModule OptixProgramGroupSingleModule::module`

Module holding single program.

7.45 OptixRelocateInput Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixBuildInputType type
- union {

```
    OptixRelocateInputInstanceArray instanceArray
    OptixRelocateInputTriangleArray triangleArray
};
```

7.45.1 Detailed Description

Relocation inputs.

See also optixAccelRelocate( )

7.45.2 Member Data Documentation

7.45.2.1

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

7.45.2.2 instanceArray

`OptixRelocateInputInstanceArray OptixRelocateInput::instanceArray`

Instance and instance pointer inputs.

7.45.2.3 triangleArray

`OptixRelocateInputTriangleArray OptixRelocateInput::triangleArray`

Triangle inputs.

7.45.2.4 type

`OptixBuildInputType OptixRelocateInput::type`

The type of the build input to relocate.

7.46 OptixRelocateInputInstanceArray Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int numInstances
- CUdeviceptr traversableHandles

7.46.1 Detailed Description

Instance and instance pointer inputs.

See also OptixRelocateInput::instanceArray

7.46.2 Member Data Documentation

7.46.2.1 numInstances

`unsigned int OptixRelocateInputInstanceArray::numInstances`

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

7.46.2.2 traversableHandles

```
CUdeviceptr OptixRelocateInputInstanceArray::traversableHandles
```

These are the traversable handles of the instances (See OptixInstance::traversableHandle) These can be used when also relocating the instances. No updates to the bounds are performed. Use optixAccelBuild to update the bounds. 'traversableHandles' may be zero when the traversables are not relocated (i.e. relocation of an IAS on the source device).

7.47 OptixRelocateInputOpacityMicromap Struct Reference

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

Public Attributes

- CUdeviceptr opacityMicromapArray

7.47.1 Member Data Documentation

7.47.1.1 opacityMicromapArray

```
CUdeviceptr OptixRelocateInputOpacityMicromap::opacityMicromapArray
```

Device pointer to a relocated opacity micromap array used by the source build input array. May be zero when no micromaps 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).

7.48 OptixRelocateInputTriangleArray Struct Reference

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

Public Attributes

- unsigned int numSbtRecords
- OptixRelocateInputOpacityMicromap opacityMicromap

7.48.1 Detailed Description

Triangle inputs.

See also OptixRelocateInput::triangleArray

7.48.2 Member Data Documentation

7.48.2.1 numSbtRecords

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

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

7.48.2.2 opacityMicromap

```
OptixRelocateInputOpacityMicromap OptixRelocateInputTriangleArray
::opacityMicromap
```

Opacity micromap inputs.

7.49 OptixRelocationInfo Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned long long info [4]

7.49.1 Detailed Description

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

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

7.49.2 Member Data Documentation

7.49.2.1 info

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

Opaque data, used internally, should not be modified.

7.50 OptixShaderBindingTable Struct Reference

`#include <optix_types.h>`

Public Attributes

- CUdeviceptr raygenRecord
- CUdeviceptr exceptionRecord

- CUdeviceptr missRecordBase
- unsigned int missRecordStrideInBytes
- unsigned int missRecordCount

- CUdeviceptr hitgroupRecordBase
- unsigned int hitgroupRecordStrideInBytes
- unsigned int hitgroupRecordCount

- CUdeviceptr callablesRecordBase
- unsigned int callablesRecordStrideInBytes
- unsigned int callablesRecordCount

7.50.1 Detailed Description

Describes the shader binding table (SBT)

See also optixLaunch( )

7.50.2 Member Data Documentation

7.50.2.1 callablesRecordBase

`CUdeviceptr OptixShaderBindingTable::callablesRecordBase`

Arrays of SBT records for callable programs. If the base address is not null, the stride and count must not be zero. If the base address is null, then the count needs to zero. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.50.2.2 callablesRecordCount

`unsigned int OptixShaderBindingTable::callablesRecordCount`

Arrays of SBT records for callable programs. If the base address is not null, the stride and count must not be zero. If the base address is null, then the count needs to zero. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.50.2.3 callablesRecordStrideInBytes

`unsigned int OptixShaderBindingTable::callablesRecordStrideInBytes`

Arrays of SBT records for callable programs. If the base address is not null, the stride and count must not be zero. If the base address is null, then the count needs to zero. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.50.2.4 exceptionRecord

`CUdeviceptr OptixShaderBindingTable::exceptionRecord`

Device address of the SBT record of the exception program. The address must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.50.2.5 hitgroupRecordBase

`CUdeviceptr OptixShaderBindingTable::hitgroupRecordBase`

Arrays of SBT records for hit groups. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.50.2.6 hitgroupRecordCount

`unsigned int OptixShaderBindingTable::hitgroupRecordCount`

Arrays of SBT records for hit groups. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.50.2.7 hitgroupRecordStrideInBytes

`unsigned int OptixShaderBindingTable::hitgroupRecordStrideInBytes`

Arrays of SBT records for hit groups. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.50.2.8 missRecordBase

`CUdeviceptr OptixShaderBindingTable::missRecordBase`

Arrays of SBT records for miss programs. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.50.2.9 missRecordCount

`unsigned int OptixShaderBindingTable::missRecordCount`

Arrays of SBT records for miss programs. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.50.2.10 missRecordStrideInBytes

`unsigned int OptixShaderBindingTable::missRecordStrideInBytes`

Arrays of SBT records for miss programs. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.50.2.11 raygenRecord

`CUdeviceptr OptixShaderBindingTable::raygenRecord`

Device address of the SBT record of the ray gen program to start launch at. The address must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.51 OptixSRTData Struct Reference

`#include <optix_types.h>`

Public Attributes

Parameters describing the SRT transformation

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

7.51.1 Detailed Description

Represents an SRT transformation.

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

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

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

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

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

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

See also OptixSRTMotionTransform::srtData, optixConvertPointerToTraversableHandle( )

7.51.2 Member Data Documentation

7.51.2.1 a

```
float OptixSRTData::a
```

7.51.2.2 b

```
float OptixSRTData::b
```

7.51.2.3 c

```
float OptixSRTData::c
```

7.51.2.4 pvx

```
float OptixSRTData::pvx
```

7.51.2.5 pvy

```
float OptixSRTData::pvy
```

7.51.2.6 pvz

```
float OptixSRTData::pvz
```

7.51.2.7 qw

```
float OptixSRTData::qw
```

7.51.2.8 qx

```
float OptixSRTData::qx
```

7.51.2.9 qy

```
float OptixSRTData::qy
```

7.51.2.10 qz

```
float OptixSRTData::qz
```

7.51.2.11 sx

```
float OptixSRTData::sx
```

7.51.2.12 sy

`float OptixSRTData::sy`

7.51.2.13 sz

`float OptixSRTData::sz`

7.51.2.14 tx

`float OptixSRTData::tx`

7.51.2.15 ty

`float OptixSRTData::ty`

7.51.2.16 tz

`float OptixSRTData::tz`

7.52 OptixSRTMotionTransform Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixTraversableHandle child
- OptixMotionOptions motionOptions
- unsigned int pad [3]
- OptixSRTData srtData [2]

7.52.1 Detailed Description

Represents an SRT motion transformation.

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

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

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

See also optixConvertPointerToTraversableHandle( )

7.52.2 Member Data Documentation

7.52.2.1 child

`OptixTraversableHandle OptixSRTMotionTransform::child`

The traversable transformed by this transformation.

7.52.2.2 motionOptions

`OptixMotionOptions OptixSRTMotionTransform::motionOptions`

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

7.52.2.3 pad

`unsigned int OptixSRTMotionTransform::pad[3]`

Padding to make the SRT data 16 byte aligned.

7.52.2.4 srtData

`OptixSRTData OptixSRTMotionTransform::srtData[2]`

The actual SRT data describing the transformation.

7.53 OptixStackSizes Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int cssRG
- unsigned int cssMS
- unsigned int cssCH
- unsigned int cssAH
- unsigned int cssIS
- unsigned int cssCC
- unsigned int dssDC

7.53.1 Detailed Description

Describes the stack size requirements of a program group.

See also optixProgramGroupGetStackSize( )

7.53.2 Member Data Documentation

7.53.2.1 cssAH

`unsigned int OptixStackSizes::cssAH`

Continuation stack size of AH programs in bytes.

7.53.2.2 cssCC

`unsigned int OptixStackSizes::cssCC`

Continuation stack size of CC programs in bytes.

7.53.2.3 cssCH

`unsigned int OptixStackSizes::cssCH`

Continuation stack size of CH programs in bytes.

7.53.2.4 cssIS

`unsigned int OptixStackSizes::cssIS`

Continuation stack size of IS programs in bytes.

7.53.2.5 cssMS

`unsigned int OptixStackSizes::cssMS`

Continuation stack size of MS programs in bytes.

7.53.2.6 cssRG

`unsigned int OptixStackSizes::cssRG`

Continuation stack size of RG programs in bytes.

7.53.2.7 dssDC

`unsigned int OptixStackSizes::dssDC`

Direct stack size of DC programs in bytes.

7.54 OptixStaticTransform Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.54.1 Detailed Description

Static transform.

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

See also optixConvertPointerToTraversableHandle( )

7.54.2 Member Data Documentation

7.54.2.1 child

`OptixTraversableHandle OptixStaticTransform::child`

The traversable transformed by this transformation.

7.54.2.2 invTransform

`float OptixStaticTransform::invTransform[12]`

Affine world-to-object transformation as 3x4 matrix in row-major layout Must be the inverse of the transform matrix.

7.54.2.3 pad

`unsigned int OptixStaticTransform::pad[2]`

Padding to make the transformations 16 byte aligned.

7.54.2.4 transform

`float OptixStaticTransform::transform[12]`

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

7.55 OptixUtilDenoiserImageTile Struct Reference

`#include <optix_denoiser_tiling.h>`

Public Attributes

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

7.55.1 Detailed Description

Tile definition.

see optixUtilDenoiserSplitImage

7.55.2 Member Data Documentation

7.55.2.1 input

`OptixImage2D OptixUtilDenoiserImageTile::input`

7.55.2.2 inputOffsetX

`unsigned int OptixUtilDenoiserImageTile::inputOffsetX`

7.55.2.3 inputOffsetY

`unsigned int OptixUtilDenoiserImageTile::inputOffsetY`

7.55.2.4 output

`OptixImage2D OptixUtilDenoiserImageTile::output`

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

`#include <optix_device_impl.h>`

8 File Documentation

8.1 optix_device_impl.h File Reference

Classes

- struct optix_internal::TypePack<... >

Namespaces

- namespace optix_internal

Macros

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

Functions

- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTraverse (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTraverse (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- static __forceinline__ __device__ void optixReorder (unsigned int coherenceHint, unsigned int numCoherenceHintBits)
- static __forceinline__ __device__ void optixReorder ()
- template<typename... Payload>
  static __forceinline__ __device__ void optixInvoke (OptixPayloadTypeID type, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixInvoke (Payload &... payload)
- template<typename... RegAttributes>
  static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int sbtOffset, unsigned int sbtStride, unsigned int instIdx, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)
- template<typename... RegAttributes>
  static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int sbtOffset, unsigned int sbtStride, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)
- template<typename... RegAttributes>
  static __forceinline__ __device__ void optixMakeHitObjectWithRecord (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int sbtRecordIndex, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int

numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)

- static __forceinline__ __device__ void optixMakeMissHitObject (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime)
- static __forceinline__ __device__ void optixMakeNopHitObject ()
- static __forceinline__ __device__ bool optixHitObjectIsHit ()
- static __forceinline__ __device__ bool optixHitObjectIsMiss ()
- static __forceinline__ __device__ bool optixHitObjectIsNop ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize ()
- static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmin ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmax ()
- static __forceinline__ __device__ float optixHitObjectGetRayTime ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex ()
- static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer ()
- static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)
- static __forceinline__ __device__ 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 optixGetMicroTriangleVertexData (float3 data[3])
- static __forceinline__ __device__ void optixGetMicroTriangleBarycentricsData (float2 data[3])
- static __forceinline__ __device__ void optixGetLinearCurveVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetCubicBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetRibbonVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ float3 optixGetRibbonNormal (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ void optixGetSphereData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle ()
- static __forceinline__ __device__ float optixGetGASMotionTimeBegin (OptixTraversableHandle handle)
- static __forceinline__ __device__ float optixGetGASMotionTimeEnd (OptixTraversableHandle handle)
- static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (OptixTraversableHandle handle)
- static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (float m[12])
- static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (float3 normal)
- static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (float3 normal)
- static __forceinline__ __device__ unsigned int optixGetTransformListSize ()

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

- static __forceinline__ __device__ unsigned int optixGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixGetHitKind ()
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsBackFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsFrontFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ()
- static __forceinline__ __device__ bool optixIsBackFaceHit ()
- static __forceinline__ __device__ bool optixIsFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleHit ()
- static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ()
- static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit ()
- static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleBackFaceHit ()
- static __forceinline__ __device__ float optixGetCurveParameter ()
- static __forceinline__ __device__ float2 optixGetRibbonParameters ()
- static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ()
- static __forceinline__ __device__ uint3 optixGetLaunchIndex ()
- static __forceinline__ __device__ uint3 optixGetLaunchDimensions ()
- static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ()
- static __forceinline__ __device__ void optixThrowException (int exceptionCode)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6, unsigned int exceptionDetail7)
- static __forceinline__ __device__ int optixGetExceptionCode ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4 ()

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

8.1.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Device side implementation

8.1.2 Macro Definition Documentation

8.1.2.1 OPTIX_DEFINE_optixGetAttribute_BODY

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

Value:

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

8.1.2.2 OPTIX_DEFINE_optixGetExceptionDetail_BODY

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

Value:

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

8.1.3 Function Documentation

8.1.3.1 optixContinuationCall()

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

8.1.3.2 optixDirectCall()

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

8.1.3.3 optixGetAttribute_0()

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

8.1.3.4 optixGetAttribute_1()

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

8.1.3.5 optixGetAttribute_2()

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

8.1.3.6 optixGetAttribute_3()

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

8.1.3.7 optixGetAttribute_4()

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

8.1.3.8 optixGetAttribute_5()

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

8.1.3.9 optixGetAttribute_6()

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

8.1.3.10 optixGetAttribute_7()

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

8.1.3.11 optixGetCatmullRomVertexData()

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

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

8.1.3.12 optixGetCubicBezierVertexData()

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

8.1.3.13 optixGetCubicBSplineVertexData()

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

8.1.3.14 optixGetCurveParameter()

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

8.1.3.15 optixGetExceptionCode()

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

8.1.3.16 optixGetExceptionDetail_0()

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

8.1.3.17 optixGetExceptionDetail_1()

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

8.1.3.18 optixGetExceptionDetail_2()

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

8.1.3.19 optixGetExceptionDetail_3()

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

8.1.3.20 optixGetExceptionDetail_4()

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

8.1.3.21 optixGetExceptionDetail_5()

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

8.1.3.22 optixGetExceptionDetail_6()

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

8.1.3.23 optixGetExceptionDetail_7()

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

8.1.3.24 optixGetExceptionLineInfo()

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

8.1.3.25 optixGetGASMotionStepCount()

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

8.1.3.26 optixGetGASMotionTimeBegin()

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

8.1.3.27 optixGetGASMotionTimeEnd()

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

8.1.3.28 optixGetGASTraversableHandle()

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

8.1.3.29 optixGetHitKind()

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

8.1.3.30 optixGetInstanceChildFromHandle()

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

8.1.3.31 optixGetInstanceId()

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

8.1.3.32 optixGetInstanceIdFromHandle()

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

8.1.3.33 optixGetInstanceIndex()

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

8.1.3.34 optixGetInstanceInverseTransformFromHandle()

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

8.1.3.35 optixGetInstanceTransformFromHandle()

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

8.1.3.36 optixGetInstanceTraversableFromIAS()

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

8.1.3.37 optixGetLaunchDimensions()

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

8.1.3.38 optixGetLaunchIndex()

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

8.1.3.39 optixGetLinearCurveVertexData()

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

8.1.3.40 optixGetMatrixMotionTransformFromHandle()

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

8.1.3.41 optixGetMicroTriangleBarycentricsData()

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

8.1.3.42 optixGetMicroTriangleVertexData()

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

8.1.3.43 optixGetObjectRayDirection()

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

8.1.3.44 optixGetObjectRayOrigin()

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

8.1.3.45 optixGetObjectToWorldTransformMatrix()

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

8.1.3.46 optixGetPayload_0()

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

8.1.3.47 optixGetPayload_1()

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

8.1.3.48 optixGetPayload_10()

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

8.1.3.49 optixGetPayload_11()

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

8.1.3.50 optixGetPayload_12()

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

8.1.3.51 optixGetPayload_13()

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

8.1.3.52 optixGetPayload_14()

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

8.1.3.53 optixGetPayload_15()

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

8.1.3.54 optixGetPayload_16()

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

8.1.3.55 optixGetPayload_17()

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

8.1.3.56 optixGetPayload_18()

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

8.1.3.57 optixGetPayload_19()

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

8.1.3.58 optixGetPayload_2()

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

8.1.3.59 optixGetPayload_20()

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

8.1.3.60 optixGetPayload_21()

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

8.1.3.61 optixGetPayload_22()

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

8.1.3.62 optixGetPayload_23()

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

8.1.3.63 optixGetPayload_24()

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

8.1.3.64 optixGetPayload_25()

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

8.1.3.65 optixGetPayload_26()

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

8.1.3.66 optixGetPayload_27()

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

8.1.3.67 optixGetPayload_28()

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

8.1.3.68 optixGetPayload_29()

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

8.1.3.69 optixGetPayload_3()

```
static __forceinline__ __device__ unsigned int optixGetPayload_3 ( ) [static]
```

8.1.3.70 optixGetPayload_30()

```
static __forceinline__ __device__ unsigned int optixGetPayload_30 ( ) [static]
```

8.1.3.71 optixGetPayload_31()

```
static __forceinline__ __device__ unsigned int optixGetPayload_31 ( ) [static]
```

8.1.3.72 optixGetPayload_4()

```
static __forceinline__ __device__ unsigned int optixGetPayload_4 ( ) [static]
```

8.1.3.73 optixGetPayload_5()

```
static __forceinline__ __device__ unsigned int optixGetPayload_5 ( ) [static]
```

8.1.3.74 optixGetPayload_6()

```
static __forceinline__ __device__ unsigned int optixGetPayload_6 ( ) [static]
```

8.1.3.75 optixGetPayload_7()

```
static __forceinline__ __device__ unsigned int optixGetPayload_7 ( ) [static]
```

8.1.3.76 optixGetPayload_8()

```
static __forceinline__ __device__ unsigned int optixGetPayload_8 ( ) [static]
```

8.1.3.77 optixGetPayload_9()

```
static __forceinline__ __device__ unsigned int optixGetPayload_9 ( ) [static]
```

8.1.3.78 optixGetPrimitiveIndex()

```
static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ( )
[static]
```

8.1.3.79 optixGetPrimitiveType() [1/2]

```
static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (
) [static]
```

8.1.3.80 optixGetPrimitiveType() [2/2]

```
static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (
          unsigned int hitKind ) [static]
```

8.1.3.81 optixGetQuadraticBSplineVertexData()

```
static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[3] ) [static]
```

8.1.3.82 optixGetRayFlags()

```
static __forceinline__ __device__ unsigned int optixGetRayFlags ( ) [static]
```

8.1.3.83 optixGetRayTime()

```
static __forceinline__ __device__ float optixGetRayTime ( ) [static]
```

8.1.3.84 optixGetRayTmax()

```
static __forceinline__ __device__ float optixGetRayTmax ( ) [static]
```

8.1.3.85 optixGetRayTmin()

```
static __forceinline__ __device__ float optixGetRayTmin ( ) [static]
```

8.1.3.86 optixGetRayVisibilityMask()

```
static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask ( )
[static]
```

8.1.3.87 optixGetRibbonNormal()

```
static __forceinline__ __device__ float3 optixGetRibbonNormal (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float2 ribbonParameters ) [static]
```

8.1.3.88 optixGetRibbonParameters()

```
static __forceinline__ __device__ float2 optixGetRibbonParameters ( ) [static]
```

8.1.3.89 optixGetRibbonVertexData()

```
static __forceinline__ __device__ void optixGetRibbonVertexData (
          OptixTraversableHandle gas,
```

```
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[3] )  [static]
```

8.1.3.90 optixGetSbtDataPointer()

```
static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ( )
[static]
```

8.1.3.91 optixGetSbtGASIndex()

```
static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ( )  [static]
```

8.1.3.92 optixGetSphereData()

```
static __forceinline__ __device__ void optixGetSphereData (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[1] )  [static]
```

8.1.3.93 optixGetSRTMotionTransformFromHandle()

```
static __forceinline__ __device__ const OptixSRTMotionTransform *
optixGetSRTMotionTransformFromHandle (
          OptixTraversableHandle handle )  [static]
```

8.1.3.94 optixGetStaticTransformFromHandle()

```
static __forceinline__ __device__ const OptixStaticTransform *
optixGetStaticTransformFromHandle (
          OptixTraversableHandle handle )  [static]
```

8.1.3.95 optixGetTransformListHandle()

```
static __forceinline__ __device__ OptixTraversableHandle
optixGetTransformListHandle (
          unsigned int index )  [static]
```

8.1.3.96 optixGetTransformListSize()

```
static __forceinline__ __device__ unsigned int optixGetTransformListSize ( )
[static]
```

8.1.3.97 optixGetTransformTypeFromHandle()

```
static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle (
          OptixTraversableHandle handle )  [static]
```

8.1.3.98 optixGetTriangleBarycentrics()

```
static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ( )
```
[static]

8.1.3.99 optixGetTriangleVertexData()

```
static __forceinline__ __device__ void optixGetTriangleVertexData (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float3 data[3] )
```
[static]

8.1.3.100 optixGetWorldRayDirection()

```
static __forceinline__ __device__ float3 optixGetWorldRayDirection ( )
```
[static]

8.1.3.101 optixGetWorldRayOrigin()

```
static __forceinline__ __device__ float3 optixGetWorldRayOrigin ( )
```
[static]

8.1.3.102 optixGetWorldToObjectTransformMatrix()

```
static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix
(
          float m[12] )
```
[static]

8.1.3.103 optixHitObjectGetAttribute_0()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0
( )
```
[static]

8.1.3.104 optixHitObjectGetAttribute_1()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1
( )
```
[static]

8.1.3.105 optixHitObjectGetAttribute_2()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2
( )
```
[static]

8.1.3.106 optixHitObjectGetAttribute_3()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3
( )
```
[static]

8.1.3.107 optixHitObjectGetAttribute_4()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4
( )
```
[static]

8.1.3.108 optixHitObjectGetAttribute_5()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5
( ) [static]
```

8.1.3.109 optixHitObjectGetAttribute_6()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6
( ) [static]
```

8.1.3.110 optixHitObjectGetAttribute_7()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7
( ) [static]
```

8.1.3.111 optixHitObjectGetHitKind()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind ( )
[static]
```

8.1.3.112 optixHitObjectGetInstanceId()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId (
) [static]
```

8.1.3.113 optixHitObjectGetInstanceIndex()

```
static __forceinline__ __device__ unsigned int
optixHitObjectGetInstanceIndex ( ) [static]
```

8.1.3.114 optixHitObjectGetPrimitiveIndex()

```
static __forceinline__ __device__ unsigned int
optixHitObjectGetPrimitiveIndex ( ) [static]
```

8.1.3.115 optixHitObjectGetRayTime()

```
static __forceinline__ __device__ float optixHitObjectGetRayTime ( ) [static]
```

8.1.3.116 optixHitObjectGetRayTmax()

```
static __forceinline__ __device__ float optixHitObjectGetRayTmax ( ) [static]
```

8.1.3.117 optixHitObjectGetRayTmin()

```
static __forceinline__ __device__ float optixHitObjectGetRayTmin ( ) [static]
```

8.1.3.118 optixHitObjectGetSbtDataPointer()

```
static __forceinline__ __device__ CUdeviceptr
optixHitObjectGetSbtDataPointer ( ) [static]
```

8.1.3.119 optixHitObjectGetSbtGASIndex()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex
( ) [static]
```

8.1.3.120 optixHitObjectGetSbtRecordIndex()

```
static __forceinline__ __device__ unsigned int
optixHitObjectGetSbtRecordIndex ( ) [static]
```

8.1.3.121 optixHitObjectGetTransformListHandle()

```
static __forceinline__ __device__ OptixTraversableHandle
optixHitObjectGetTransformListHandle (
          unsigned int index ) [static]
```

8.1.3.122 optixHitObjectGetTransformListSize()

```
static __forceinline__ __device__ unsigned int
optixHitObjectGetTransformListSize ( ) [static]
```

8.1.3.123 optixHitObjectGetWorldRayDirection()

```
static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection
( ) [static]
```

8.1.3.124 optixHitObjectGetWorldRayOrigin()

```
static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin ( )
[static]
```

8.1.3.125 optixHitObjectIsHit()

```
static __forceinline__ __device__ bool optixHitObjectIsHit ( ) [static]
```

8.1.3.126 optixHitObjectIsMiss()

```
static __forceinline__ __device__ bool optixHitObjectIsMiss ( ) [static]
```

8.1.3.127 optixHitObjectIsNop()

```
static __forceinline__ __device__ bool optixHitObjectIsNop ( ) [static]
```

8.1.3.128 optixIgnoreIntersection()

```
static __forceinline__ __device__ void optixIgnoreIntersection ( ) [static]
```

8.1.3.129 optixInvoke() [1/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixInvoke (
          OptixPayloadTypeID type,
          Payload &... payload ) [static]
```

8.1.3.130 optixInvoke() [2/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixInvoke (
          Payload &... payload ) [static]
```

8.1.3.131 optixIsBackFaceHit() [1/2]

```
static __forceinline__ __device__ bool optixIsBackFaceHit ( ) [static]
```

8.1.3.132 optixIsBackFaceHit() [2/2]

```
static __forceinline__ __device__ bool optixIsBackFaceHit (
          unsigned int hitKind ) [static]
```

8.1.3.133 optixIsDisplacedMicromeshTriangleBackFaceHit()

```
static __forceinline__ __device__ bool
optixIsDisplacedMicromeshTriangleBackFaceHit ( ) [static]
```

8.1.3.134 optixIsDisplacedMicromeshTriangleFrontFaceHit()

```
static __forceinline__ __device__ bool
optixIsDisplacedMicromeshTriangleFrontFaceHit ( ) [static]
```

8.1.3.135 optixIsDisplacedMicromeshTriangleHit()

```
static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit
( ) [static]
```

8.1.3.136 optixIsFrontFaceHit() [1/2]

```
static __forceinline__ __device__ bool optixIsFrontFaceHit ( ) [static]
```

8.1.3.137 optixIsFrontFaceHit() [2/2]

```
static __forceinline__ __device__ bool optixIsFrontFaceHit (
          unsigned int hitKind ) [static]
```

8.1.3.138 optixIsTriangleBackFaceHit()

```
static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ( ) [static]
```

8.1.3.139 optixIsTriangleFrontFaceHit()

```
static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ( ) [static]
```

8.1.3.140 optixIsTriangleHit()

```
static __forceinline__ __device__ bool optixIsTriangleHit ( ) [static]
```

8.1.3.141 optixMakeHitObject() [1/2]

```
template<typename... RegAttributes>
static __forceinline__ __device__ void optixMakeHitObject (
          OptixTraversableHandle handle,
          float3 rayOrigin,
          float3 rayDirection,
          float tmin,
          float tmax,
```

```
            float rayTime,
            unsigned int sbtOffset,
            unsigned int sbtStride,
            unsigned int instIdx,
            const OptixTraversableHandle * transforms,
            unsigned int numTransforms,
            unsigned int sbtGASIdx,
            unsigned int primIdx,
            unsigned int hitKind,
            RegAttributes... regAttributes ) [static]
```

8.1.3.142 optixMakeHitObject() [2/2]

```
template<typename... RegAttributes>
static __forceinline__ __device__ void optixMakeHitObject (
            OptixTraversableHandle handle,
            float3 rayOrigin,
            float3 rayDirection,
            float tmin,
            float tmax,
            float rayTime,
            unsigned int sbtOffset,
            unsigned int sbtStride,
            unsigned int instIdx,
            unsigned int sbtGASIdx,
            unsigned int primIdx,
            unsigned int hitKind,
            RegAttributes... regAttributes ) [static]
```

8.1.3.143 optixMakeHitObjectWithRecord()

```
template<typename... RegAttributes>
static __forceinline__ __device__ void optixMakeHitObjectWithRecord (
            OptixTraversableHandle handle,
            float3 rayOrigin,
            float3 rayDirection,
            float tmin,
            float tmax,
            float rayTime,
            unsigned int sbtRecordIndex,
            unsigned int instIdx,
            const OptixTraversableHandle * transforms,
            unsigned int numTransforms,
            unsigned int sbtGASIdx,
```

```
          unsigned int primIdx,
          unsigned int hitKind,
          RegAttributes... regAttributes ) [static]
```

8.1.3.144 optixMakeMissHitObject()

```
static __forceinline__ __device__ void optixMakeMissHitObject (
          unsigned int missSBTIndex,
          float3 rayOrigin,
          float3 rayDirection,
          float tmin,
          float tmax,
          float rayTime ) [static]
```

8.1.3.145 optixMakeNopHitObject()

```
static __forceinline__ __device__ void optixMakeNopHitObject ( ) [static]
```

8.1.3.146 optixReorder() [1/2]

```
static __forceinline__ __device__ void optixReorder ( ) [static]
```

8.1.3.147 optixReorder() [2/2]

```
static __forceinline__ __device__ void optixReorder (
          unsigned int coherenceHint,
          unsigned int numCoherenceHintBits ) [static]
```

8.1.3.148 optixReportIntersection() [1/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
          float hitT,
          unsigned int hitKind ) [static]
```

8.1.3.149 optixReportIntersection() [2/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
          float hitT,
          unsigned int hitKind,
          unsigned int a0 ) [static]
```

8.1.3.150 optixReportIntersection() [3/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
          float hitT,
          unsigned int hitKind,
          unsigned int a0,
          unsigned int a1 ) [static]
```

8.1.3.151 optixReportIntersection() [4/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
          float hitT,
          unsigned int hitKind,
          unsigned int a0,
          unsigned int a1,
          unsigned int a2 )  [static]
```

8.1.3.152 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]
```

8.1.3.153 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]
```

8.1.3.154 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]
```

8.1.3.155 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]
```

8.1.3.156 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]
```

8.1.3.157 optixSetPayload_0()

```
static __forceinline__ __device__ void optixSetPayload_0 (
          unsigned int p )  [static]
```

8.1.3.158 optixSetPayload_1()

```
static __forceinline__ __device__ void optixSetPayload_1 (
          unsigned int p )  [static]
```

8.1.3.159 optixSetPayload_10()

```
static __forceinline__ __device__ void optixSetPayload_10 (
          unsigned int p )  [static]
```

8.1.3.160 optixSetPayload_11()

```
static __forceinline__ __device__ void optixSetPayload_11 (
          unsigned int p )  [static]
```

8.1.3.161 optixSetPayload_12()

```
static __forceinline__ __device__ void optixSetPayload_12 (
          unsigned int p )  [static]
```

8.1.3.162 optixSetPayload_13()

```
static __forceinline__ __device__ void optixSetPayload_13 (
          unsigned int p ) [static]
```

8.1.3.163 optixSetPayload_14()

```
static __forceinline__ __device__ void optixSetPayload_14 (
          unsigned int p ) [static]
```

8.1.3.164 optixSetPayload_15()

```
static __forceinline__ __device__ void optixSetPayload_15 (
          unsigned int p ) [static]
```

8.1.3.165 optixSetPayload_16()

```
static __forceinline__ __device__ void optixSetPayload_16 (
          unsigned int p ) [static]
```

8.1.3.166 optixSetPayload_17()

```
static __forceinline__ __device__ void optixSetPayload_17 (
          unsigned int p ) [static]
```

8.1.3.167 optixSetPayload_18()

```
static __forceinline__ __device__ void optixSetPayload_18 (
          unsigned int p ) [static]
```

8.1.3.168 optixSetPayload_19()

```
static __forceinline__ __device__ void optixSetPayload_19 (
          unsigned int p ) [static]
```

8.1.3.169 optixSetPayload_2()

```
static __forceinline__ __device__ void optixSetPayload_2 (
          unsigned int p ) [static]
```

8.1.3.170 optixSetPayload_20()

```
static __forceinline__ __device__ void optixSetPayload_20 (
          unsigned int p ) [static]
```

8.1.3.171 optixSetPayload_21()

```
static __forceinline__ __device__ void optixSetPayload_21 (
          unsigned int p ) [static]
```

8.1.3.172 optixSetPayload_22()

```
static __forceinline__ __device__ void optixSetPayload_22 (
```

```
          unsigned int p ) [static]
```

8.1.3.173 optixSetPayload_23()

```
static __forceinline__ __device__ void optixSetPayload_23 (
          unsigned int p ) [static]
```

8.1.3.174 optixSetPayload_24()

```
static __forceinline__ __device__ void optixSetPayload_24 (
          unsigned int p ) [static]
```

8.1.3.175 optixSetPayload_25()

```
static __forceinline__ __device__ void optixSetPayload_25 (
          unsigned int p ) [static]
```

8.1.3.176 optixSetPayload_26()

```
static __forceinline__ __device__ void optixSetPayload_26 (
          unsigned int p ) [static]
```

8.1.3.177 optixSetPayload_27()

```
static __forceinline__ __device__ void optixSetPayload_27 (
          unsigned int p ) [static]
```

8.1.3.178 optixSetPayload_28()

```
static __forceinline__ __device__ void optixSetPayload_28 (
          unsigned int p ) [static]
```

8.1.3.179 optixSetPayload_29()

```
static __forceinline__ __device__ void optixSetPayload_29 (
          unsigned int p ) [static]
```

8.1.3.180 optixSetPayload_3()

```
static __forceinline__ __device__ void optixSetPayload_3 (
          unsigned int p ) [static]
```

8.1.3.181 optixSetPayload_30()

```
static __forceinline__ __device__ void optixSetPayload_30 (
          unsigned int p ) [static]
```

8.1.3.182 optixSetPayload_31()

```
static __forceinline__ __device__ void optixSetPayload_31 (
          unsigned int p ) [static]
```

8.1.3.183 optixSetPayload_4()

```
static __forceinline__ __device__ void optixSetPayload_4 (
          unsigned int p ) [static]
```

8.1.3.184 optixSetPayload_5()

```
static __forceinline__ __device__ void optixSetPayload_5 (
          unsigned int p ) [static]
```

8.1.3.185 optixSetPayload_6()

```
static __forceinline__ __device__ void optixSetPayload_6 (
          unsigned int p ) [static]
```

8.1.3.186 optixSetPayload_7()

```
static __forceinline__ __device__ void optixSetPayload_7 (
          unsigned int p ) [static]
```

8.1.3.187 optixSetPayload_8()

```
static __forceinline__ __device__ void optixSetPayload_8 (
          unsigned int p ) [static]
```

8.1.3.188 optixSetPayload_9()

```
static __forceinline__ __device__ void optixSetPayload_9 (
          unsigned int p ) [static]
```

8.1.3.189 optixSetPayloadTypes()

```
static __forceinline__ __device__ void optixSetPayloadTypes (
          unsigned int types ) [static]
```

8.1.3.190 optixTerminateRay()

```
static __forceinline__ __device__ void optixTerminateRay ( ) [static]
```

8.1.3.191 optixTexFootprint2D()

```
static __forceinline__ __device__ uint4 optixTexFootprint2D (
          unsigned long long tex,
          unsigned int texInfo,
          float x,
          float y,
          unsigned int * singleMipLevel ) [static]
```

8.1.3.192 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]
```

8.1.3.193 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]
```

8.1.3.194 optixThrowException() [1/9]

```
static __forceinline__ __device__ void optixThrowException (
          int exceptionCode ) [static]
```

8.1.3.195 optixThrowException() [2/9]

```
static __forceinline__ __device__ void optixThrowException (
          int exceptionCode,
          unsigned int exceptionDetail0 ) [static]
```

8.1.3.196 optixThrowException() [3/9]

```
static __forceinline__ __device__ void optixThrowException (
          int exceptionCode,
          unsigned int exceptionDetail0,
          unsigned int exceptionDetail1 ) [static]
```

8.1.3.197 optixThrowException() [4/9]

```
static __forceinline__ __device__ void optixThrowException (
          int exceptionCode,
          unsigned int exceptionDetail0,
          unsigned int exceptionDetail1,
          unsigned int exceptionDetail2 ) [static]
```

8.1.3.198 optixThrowException() [5/9]

```
static __forceinline__ __device__ void optixThrowException (
          int exceptionCode,
          unsigned int exceptionDetail0,
          unsigned int exceptionDetail1,
          unsigned int exceptionDetail2,
          unsigned int exceptionDetail3 ) [static]
```

8.1.3.199 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]
```

8.1.3.200 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]
```

8.1.3.201 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]
```

8.1.3.202 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]
```

8.1.3.203 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]
```

8.1.3.204 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]
```

8.1.3.205 optixTransformNormalFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixTransformNormalFromObjectToWorldSpace (
          float3 normal ) [static]
```

8.1.3.206 optixTransformNormalFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixTransformNormalFromWorldToObjectSpace (
          float3 normal ) [static]
```

8.1.3.207 optixTransformPointFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixTransformPointFromObjectToWorldSpace (
          float3 point ) [static]
```

8.1.3.208 optixTransformPointFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixTransformPointFromWorldToObjectSpace (
          float3 point ) [static]
```

8.1.3.209 optixTransformVectorFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixTransformVectorFromObjectToWorldSpace (
          float3 vec ) [static]
```

8.1.3.210 optixTransformVectorFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixTransformVectorFromWorldToObjectSpace (
          float3 vec ) [static]
```

8.1.3.211 optixTraverse() [1/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixTraverse (
          OptixPayloadTypeID type,
          OptixTraversableHandle handle,
          float3 rayOrigin,
          float3 rayDirection,
          float tmin,
          float tmax,
          float rayTime,
          OptixVisibilityMask visibilityMask,
          unsigned int rayFlags,
```

```
            unsigned int SBToffset,
            unsigned int SBTstride,
            unsigned int missSBTIndex,
            Payload &... payload ) [static]
```

8.1.3.212 optixTraverse() [2/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixTraverse (
            OptixTraversableHandle handle,
            float3 rayOrigin,
            float3 rayDirection,
            float tmin,
            float tmax,
            float rayTime,
            OptixVisibilityMask visibilityMask,
            unsigned int rayFlags,
            unsigned int SBToffset,
            unsigned int SBTstride,
            unsigned int missSBTIndex,
            Payload &... payload ) [static]
```

8.1.3.213 optixUndefinedValue()

```
static __forceinline__ __device__ unsigned int optixUndefinedValue ( ) [static]
```

8.2 optix_device_impl.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2023 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_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_DEVICE_IMPL_H
```

```
34 #define OPTIX_OPTIX_DEVICE_IMPL_H
35
36 #include "internal/optix_device_impl_transformations.h"
37
38 #ifndef __CUDACC_RTC__
39 #include <initializer_list>
40 #include <type_traits>
41 #endif
42
43 namespace optix_internal {
44 template <typename...>
45 struct TypePack{};
46 }  // namespace optix_internal
47
48 template <typename... Payload>
49 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
50                                                   float3                rayOrigin,
51                                                   float3                rayDirection,
52                                                   float                 tmin,
53                                                   float                 tmax,
54                                                   float                 rayTime,
55                                                   OptixVisibilityMask   visibilityMask,
56                                                   unsigned int          rayFlags,
57                                                   unsigned int          SBToffset,
58                                                   unsigned int          SBTstride,
59                                                   unsigned int          missSBTIndex,
60                                                   Payload&...             payload)
61 {
62     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
63     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
64     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
65     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
66 #ifndef __CUDACC_RTC__
67     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int»::value,
68                   "All payload parameters need to be unsigned int.");
69 #endif
70
71     OptixPayloadTypeID type = OPTIX_PAYLOAD_TYPE_DEFAULT;
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"(type), "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 optixTraverse(OptixTraversableHandle handle,
102                                                      float3                 rayOrigin,
103                                                      float3                 rayDirection,
104                                                      float                  tmin,
105                                                      float                  tmax,
106                                                      float                  rayTime,
107                                                      OptixVisibilityMask    visibilityMask,
108                                                      unsigned int           rayFlags,
109                                                      unsigned int           SBToffset,
110                                                      unsigned int           SBTstride,
111                                                      unsigned int           missSBTIndex,
112                                                      Payload&... payload)
113 {
114     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
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 #ifndef __CUDACC_RTC__
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 #endif
122
123     OptixPayloadTypeID type = OPTIX_PAYLOAD_TYPE_DEFAULT;
124     float              ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
125     float              dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
126     unsigned int p[33]       = {0, payload...};
127     int          payloadSize = (int)sizeof...(Payload);
128     asm volatile(
129         "call"
130
"(%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,%"
131         "29,%30,%31),"
132         "_optix_hitobject_traverse,"
133
"(%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,
134         "59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
135         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
136           "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
137           "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
138           "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
139           "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
140         : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
141           "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBToffset), "r"(SBTstride),
142           "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
143           "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
144           "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
145           "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
146           "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
147         );
148     unsigned int index = 1;
149     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
150 }
151
152 template <typename... Payload>
153 static __forceinline__ __device__ void optixTrace(OptixPayloadTypeID     type,
154                                                   OptixTraversableHandle handle,
155                                                   float3                 rayOrigin,
156                                                   float3                 rayDirection,
157                                                   float                  tmin,
158                                                   float                  tmax,
159                                                   float                  rayTime,
160                                                   OptixVisibilityMask    visibilityMask,
161                                                   unsigned int           rayFlags,
```

```
162                                                 unsigned int           SBToffset,
163                                                 unsigned int           SBTstride,
164                                                 unsigned int           missSBTIndex,
165                                                 Payload&...              payload)
166 {
167     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
168     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
169     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
170     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
171 #ifndef __CUDACC_RTC__
172     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int»::value,
173                   "All payload parameters need to be unsigned int.");
174 #endif
175
176     float        ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
177     float        dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
178     unsigned int p[33]       = {0, payload...};
179     int          payloadSize = (int)sizeof...(Payload);
180
181     asm volatile(
182         "call"
183
"(%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,%"
184         "29,%30,%31),"
185         "_optix_trace_typed_32,"
186
"(%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,
187         "59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
188         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
189           "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
190           "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
191           "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
192           "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
193         : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
194           "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBToffset), "r"(SBTstride),
195           "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
196           "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
197           "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
198           "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
199           "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
200         :);
201     unsigned int index = 1;
202     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
203 }
204
205 template <typename... Payload>
206 static __forceinline__ __device__ void optixTraverse(OptixPayloadTypeID     type,
207                                                      OptixTraversableHandle handle,
208                                                      float3                 rayOrigin,
209                                                      float3                 rayDirection,
210                                                      float                  tmin,
211                                                      float                  tmax,
212                                                      float                  rayTime,
213                                                      OptixVisibilityMask    visibilityMask,
214                                                      unsigned int           rayFlags,
215                                                      unsigned int           SBToffset,
216                                                      unsigned int           SBTstride,
217                                                      unsigned int           missSBTIndex,
218                                                      Payload&... payload)
219 {
220     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
221     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
222     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
223     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
224 #ifndef __CUDACC_RTC__
225     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
```

```
optix_internal::TypePack<Payload..., unsigned int>::value,
                   "All payload parameters need to be unsigned int.");
#endif

    float        ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
    float        dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
    unsigned int p[33]       = {0, payload...};
    int          payloadSize = (int)sizeof...(Payload);
    asm volatile(
        "call"

"(%0,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18,%19,%20,%21,%22,%23,%24,%25,%26,%27,%28,%"
        "29,%30,%31),"
        "_optix_hitobject_traverse,"

"(%32,%33,%34,%35,%36,%37,%38,%39,%40,%41,%42,%43,%44,%45,%46,%47,%48,%49,%50,%51,%52,%53,%54,%55,%56,%57,%58,
        "59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
        : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
          "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
          "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
          "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
          "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
        : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
          "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBToffset), "r"(SBTstride),
          "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
          "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
          "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
          "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
          "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
        );
    unsigned int index = 1;
    (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
}

static __forceinline__ __device__ void optixReorder(unsigned int coherenceHint, unsigned int
numCoherenceHintBits)
{
    asm volatile(
         "call"
         "(),"
         "_optix_hitobject_reorder,"
         "(%0,%1);"
         :
         : "r"(coherenceHint), "r"(numCoherenceHintBits)
         );
}

static __forceinline__ __device__ void optixReorder()
{
    unsigned int coherenceHint = 0;
    unsigned int numCoherenceHintBits = 0;
    asm volatile(
         "call"
         "(),"
         "_optix_hitobject_reorder,"
         "(%0,%1);"
         :
         : "r"(coherenceHint), "r"(numCoherenceHintBits)
         );
}

template <typename... Payload>
static __forceinline__ __device__ void optixInvoke(OptixPayloadTypeID type, Payload&... payload)
{
    // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
    // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
    // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
```

```
289     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
290 #ifndef __CUDACC_RTC__
291     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int»::value,
292                   "All payload parameters need to be unsigned int.");
293 #endif
294
295     unsigned int p[33]       = {0, payload...};
296     int          payloadSize = (int)sizeof...(Payload);
297
298     asm volatile(
299         "call"
300
"(%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,%"
301         "29,%30,%31),"
302         "_optix_hitobject_invoke,"
303
"(%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,
304         "59,%60,%61,%62,%63,%64,%65);"
305         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
306           "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
307           "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
308           "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
309           "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
310         : "r"(type), "r"(payloadSize), "r"(p[1]), "r"(p[2]),
311           "r"(p[3]), "r"(p[4]), "r"(p[5]), "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]),
312           "r"(p[11]), "r"(p[12]), "r"(p[13]), "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]),
313           "r"(p[18]), "r"(p[19]), "r"(p[20]), "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]),
314           "r"(p[25]), "r"(p[26]), "r"(p[27]), "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
315         :);
316
317     unsigned int index = 1;
318     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
319 }
320
321 template <typename... Payload>
322 static __forceinline__ __device__ void optixInvoke(Payload&... payload)
323 {
324     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
325     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
326     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
327     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
328 #ifndef __CUDACC_RTC__
329     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int»::value,
330                   "All payload parameters need to be unsigned int.");
331 #endif
332
333     OptixPayloadTypeID type        = OPTIX_PAYLOAD_TYPE_DEFAULT;
334     unsigned int       p[33]       = {0, payload...};
335     int                payloadSize = (int)sizeof...(Payload);
336
337     asm volatile(
338         "call"
339
"(%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,%"
340         "29,%30,%31),"
341         "_optix_hitobject_invoke,"
342
"(%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,
343         "59,%60,%61,%62,%63,%64,%65);"
344         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
345           "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
346           "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
347           "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
348           "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
349         : "r"(type), "r"(payloadSize), "r"(p[1]), "r"(p[2]),
```

```
350           "r"(p[3]), "r"(p[4]), "r"(p[5]), "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]),
351           "r"(p[11]), "r"(p[12]), "r"(p[13]), "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]),
352           "r"(p[18]), "r"(p[19]), "r"(p[20]), "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]),
353           "r"(p[25]), "r"(p[26]), "r"(p[27]), "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
354         :);
355
356     unsigned int index = 1;
357     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
358 }
359
360 template <typename... RegAttributes>
361 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle handle,
362                                                           float3                 rayOrigin,
363                                                           float3                 rayDirection,
364                                                           float                  tmin,
365                                                           float                  tmax,
366                                                           float                  rayTime,
367                                                           unsigned int           sbtOffset,
368                                                           unsigned int           sbtStride,
369                                                           unsigned int           instIdx,
370                                                           unsigned int           sbtGASIdx,
371                                                           unsigned int           primIdx,
372                                                           unsigned int           hitKind,
373                                                           RegAttributes... regAttributes)
374 {
375     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
376     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
377     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
378     static_assert(sizeof...(RegAttributes) <= 8, "Only up to 8 register attribute values are allowed.");
379 #ifndef __CUDACC_RTC__
380     static_assert(
381         std::is_same<optix_internal::TypePack<unsigned int, RegAttributes...>,
optix_internal::TypePack<RegAttributes..., unsigned int»::value,
382         "All register attribute parameters need to be unsigned int.");
383 #endif
384
385     float        ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
386     float        dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
387     unsigned int a[9]     = {0, regAttributes...};
388     int          attrSize = (int)sizeof...(RegAttributes);
389
390     OptixTraversableHandle* transforms    = nullptr;
391     unsigned int            numTransforms = 0;
392
393     asm volatile(
394          "call"
395          "(),"
396          "_optix_hitobject_make_hit,"
397
"(%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);"
398          :
399          : "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin), "f"(tmax),
400            "f"(rayTime), "r"(sbtOffset), "r"(sbtStride), "r"(instIdx), "l"(transforms),
"r"(numTransforms),
401            "r"(sbtGASIdx), "r"(primIdx), "r"(hitKind), "r"(attrSize), "r"(a[1]), "r"(a[2]), "r"(a[3]),
402            "r"(a[4]), "r"(a[5]), "r"(a[6]), "r"(a[7]), "r"(a[8])
403          :);
404 }
405
406 template <typename... RegAttributes>
407 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle        handle,
408                                                           float3                         rayOrigin,
409                                                           float3                         rayDirection,
410                                                           float                          tmin,
411                                                           float                          tmax,
412                                                           float                          rayTime,
413                                                           unsigned int                   sbtOffset,
```

```
414                                                     unsigned int                  sbtStride,
415                                                     unsigned int                  instIdx,
416                                                     const OptixTraversableHandle* transforms,
417                                                     unsigned int                  numTransforms,
418                                                     unsigned int                  sbtGASIdx,
419                                                     unsigned int                  primIdx,
420                                                     unsigned int                  hitKind,
421                                                     RegAttributes... regAttributes)
422 {
423     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
424     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
425     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
426     static_assert(sizeof...(RegAttributes) <= 8, "Only up to 8 register attribute values are allowed.");
427 #ifndef __CUDACC_RTC__
428     static_assert(
429         std::is_same<optix_internal::TypePack<unsigned int, RegAttributes...>,
optix_internal::TypePack<RegAttributes..., unsigned int»::value,
430         "All register attribute parameters need to be unsigned int.");
431 #endif
432
433     float        ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
434     float        dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
435     unsigned int a[9]     = {0, regAttributes...};
436     int          attrSize = (int)sizeof...(RegAttributes);
437
438     asm volatile(
439          "call"
440          "(),"
441          "_optix_hitobject_make_hit,"
442
"(%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);"
443          :
444          : "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin), "f"(tmax),
445            "f"(rayTime), "r"(sbtOffset), "r"(sbtStride), "r"(instIdx), "l"(transforms),
"r"(numTransforms),
446            "r"(sbtGASIdx), "r"(primIdx), "r"(hitKind), "r"(attrSize), "r"(a[1]), "r"(a[2]), "r"(a[3]),
447            "r"(a[4]), "r"(a[5]), "r"(a[6]), "r"(a[7]), "r"(a[8])
448          :);
449 }
450
451 template <typename... RegAttributes>
452 static __forceinline__ __device__ void optixMakeHitObjectWithRecord(OptixTraversableHandle       handle,
453                                                                      float3                     rayOrigin,
454                                                                      float3                     rayDirection,
455                                                                      float                      tmin,
456                                                                      float                      tmax,
457                                                                      float                      rayTime,
458                                                                      unsigned int               sbtRecordIndex,
459                                                                      unsigned int               instIdx,
460                                                                      const OptixTraversableHandle* transforms,
461                                                                      unsigned int               numTransforms,
462                                                                      unsigned int               sbtGASIdx,
463                                                                      unsigned int               primIdx,
464                                                                      unsigned int               hitKind,
465                                                                      RegAttributes... regAttributes)
466 {
467     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
468     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
469     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
470     static_assert(sizeof...(RegAttributes) <= 8, "Only up to 8 register attribute values are allowed.");
471 #ifndef __CUDACC_RTC__
472     static_assert(
473         std::is_same<optix_internal::TypePack<unsigned int, RegAttributes...>,
optix_internal::TypePack<RegAttributes..., unsigned int»::value,
474         "All register attribute parameters need to be unsigned int.");
475 #endif
476
```

```
477     float        ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
478     float        dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
479     unsigned int a[9]     = {0, regAttributes...};
480     int          attrSize = (int)sizeof...(RegAttributes);
481
482     asm volatile(
483          "call"
484          "(),"
485          "_optix_hitobject_make_hit_with_record,"
486
"(%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);"
487          :
488          : "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin), "f"(tmax),
489            "f"(rayTime), "r"(sbtRecordIndex), "r"(instIdx), "l"(transforms), "r"(numTransforms),
490            "r"(sbtGASIdx), "r"(primIdx), "r"(hitKind), "r"(attrSize), "r"(a[1]), "r"(a[2]), "r"(a[3]),
491            "r"(a[4]), "r"(a[5]), "r"(a[6]), "r"(a[7]), "r"(a[8])
492          :);
493 }
494
495 static __forceinline__ __device__ void optixMakeMissHitObject(unsigned int missSBTIndex,
496                                                                float3       rayOrigin,
497                                                                float3       rayDirection,
498                                                                float        tmin,
499                                                                float        tmax,
500                                                                float        rayTime)
501 {
502     float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
503     float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
504
505     asm volatile(
506          "call"
507          "(),"
508          "_optix_hitobject_make_miss,"
509          "(%0,%1,%2,%3,%4,%5,%6,%7,%8,%9);"
510          :
511          : "r"(missSBTIndex), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
512            "f"(tmax), "f"(rayTime)
513          :);
514 }
515
516 static __forceinline__ __device__ void optixMakeNopHitObject()
517 {
518     asm volatile(
519          "call"
520          "(),"
521          "_optix_hitobject_make_nop,"
522          "();"
523          :
524          :
525          :);
526 }
527
528 static __forceinline__ __device__ bool optixHitObjectIsHit()
529 {
530     unsigned int result;
531     asm volatile(
532          "call (%0), _optix_hitobject_is_hit,"
533          "();"
534          : "=r"(result)
535          :
536          :);
537     return result;
538 }
539
540 static __forceinline__ __device__ bool optixHitObjectIsMiss()
541 {
542     unsigned int result;
```

```
543     asm volatile(
544         "call (%0), _optix_hitobject_is_miss,"
545         "();"
546         : "=r"(result)
547         :
548         :);
549     return result;
550 }
551
552 static __forceinline__ __device__ bool optixHitObjectIsNop()
553 {
554     unsigned int result;
555     asm volatile(
556         "call (%0), _optix_hitobject_is_nop,"
557         "();"
558         : "=r"(result)
559         :
560         :);
561     return result;
562 }
563
564 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId()
565 {
566     unsigned int result;
567     asm volatile(
568         "call (%0), _optix_hitobject_get_instance_id,"
569         "();"
570         : "=r"(result)
571         :
572         :);
573     return result;
574 }
575
576 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex()
577 {
578     unsigned int result;
579     asm volatile(
580         "call (%0), _optix_hitobject_get_instance_idx,"
581         "();"
582         : "=r"(result)
583         :
584         :);
585     return result;
586 }
587
588 static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex()
589 {
590     unsigned int result;
591     asm volatile(
592         "call (%0), _optix_hitobject_get_primitive_idx,"
593         "();"
594         : "=r"(result)
595         :
596         :);
597     return result;
598 }
599
600 static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize()
601 {
602     unsigned int result;
603     asm volatile(
604         "call (%0), _optix_hitobject_get_transform_list_size,"
605         "();"
606         : "=r"(result)
607         :
608         :);
609     return result;
```

```
610 }
611
612 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle(unsigned
int index)
613 {
614     unsigned long long result;
615     asm volatile(
616          "call (%0), _optix_hitobject_get_transform_list_handle,"
617          "(%1);"
618          : "=l"(result)
619          : "r"(index)
620          :);
621     return result;
622 }
623
624 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex()
625 {
626     unsigned int result;
627     asm volatile(
628          "call (%0), _optix_hitobject_get_sbt_gas_idx,"
629          "();"
630          : "=r"(result)
631          :
632          :);
633     return result;
634 }
635
636 static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind()
637 {
638     unsigned int result;
639     asm volatile(
640          "call (%0), _optix_hitobject_get_hitkind,"
641          "();"
642          : "=r"(result)
643          :
644          :);
645     return result;
646 }
647
648 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin()
649 {
650     float x, y, z;
651     asm volatile(
652          "call (%0), _optix_hitobject_get_world_ray_origin_x,"
653          "();"
654          : "=f"(x)
655          :
656          :);
657     asm volatile(
658          "call (%0), _optix_hitobject_get_world_ray_origin_y,"
659          "();"
660          : "=f"(y)
661          :
662          :);
663     asm volatile(
664          "call (%0), _optix_hitobject_get_world_ray_origin_z,"
665          "();"
666          : "=f"(z)
667          :
668          :);
669     return make_float3(x, y, z);
670 }
671
672 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection()
673 {
674     float x, y, z;
675     asm volatile(
```

```
676          "call (%0), _optix_hitobject_get_world_ray_direction_x,"
677          "();"
678          : "=f"(x)
679          :
680          :);
681     asm volatile(
682          "call (%0), _optix_hitobject_get_world_ray_direction_y,"
683          "();"
684          : "=f"(y)
685          :
686          :);
687     asm volatile(
688          "call (%0), _optix_hitobject_get_world_ray_direction_z,"
689          "();"
690          : "=f"(z)
691          :
692          :);
693     return make_float3(x, y, z);
694 }
695
696 static __forceinline__ __device__ float optixHitObjectGetRayTmin()
697 {
698     float result;
699     asm volatile(
700          "call (%0), _optix_hitobject_get_ray_tmin,"
701          "();"
702          : "=f"(result)
703          :
704          :);
705     return result;
706 }
707
708 static __forceinline__ __device__ float optixHitObjectGetRayTmax()
709 {
710     float result;
711     asm volatile(
712          "call (%0), _optix_hitobject_get_ray_tmax,"
713          "();"
714          : "=f"(result)
715          :
716          :);
717     return result;
718 }
719
720 static __forceinline__ __device__ float optixHitObjectGetRayTime()
721 {
722     float result;
723     asm volatile(
724          "call (%0), _optix_hitobject_get_ray_time,"
725          "();"
726          : "=f"(result)
727          :
728          :);
729     return result;
730 }
731
732 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0()
733 {
734     unsigned int ret;
735     asm volatile(
736          "call (%0), _optix_hitobject_get_attribute,"
737          "(%1);"
738          : "=r"(ret)
739          : "r"(0)
740          :);
741     return ret;
742 }
```

```
743
744 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1()
745 {
746     unsigned int ret;
747     asm volatile(
748          "call (%0), _optix_hitobject_get_attribute,"
749          "(%1);"
750          : "=r"(ret)
751          : "r"(1)
752          :);
753     return ret;
754 }
755
756 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2()
757 {
758     unsigned int ret;
759     asm volatile(
760          "call (%0), _optix_hitobject_get_attribute,"
761          "(%1);"
762          : "=r"(ret)
763          : "r"(2)
764          :);
765     return ret;
766 }
767
768 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3()
769 {
770     unsigned int ret;
771     asm volatile(
772          "call (%0), _optix_hitobject_get_attribute,"
773          "(%1);"
774          : "=r"(ret)
775          : "r"(3)
776          :);
777     return ret;
778 }
779
780 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4()
781 {
782     unsigned int ret;
783     asm volatile(
784          "call (%0), _optix_hitobject_get_attribute,"
785          "(%1);"
786          : "=r"(ret)
787          : "r"(4)
788          :);
789     return ret;
790 }
791
792 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5()
793 {
794     unsigned int ret;
795     asm volatile(
796          "call (%0), _optix_hitobject_get_attribute,"
797          "(%1);"
798          : "=r"(ret)
799          : "r"(5)
800          :);
801     return ret;
802 }
803
804 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6()
805 {
806     unsigned int ret;
807     asm volatile(
808          "call (%0), _optix_hitobject_get_attribute,"
809          "(%1);"
```

```
810           : "=r"(ret)
811           : "r"(6)
812           :);
813     return ret;
814 }
815
816 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7()
817 {
818     unsigned int ret;
819     asm volatile(
820           "call (%0), _optix_hitobject_get_attribute,"
821           "(%1);"
822           : "=r"(ret)
823           : "r"(7)
824           :);
825     return ret;
826 }
827
828 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex()
829 {
830     unsigned int result;
831     asm volatile(
832           "call (%0), _optix_hitobject_get_sbt_record_index,"
833           "();"
834           : "=r"(result)
835           :
836           :);
837     return result;
838 }
839
840 static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer()
841 {
842     unsigned long long ptr;
843     asm volatile(
844           "call (%0), _optix_hitobject_get_sbt_data_pointer,"
845           "();"
846           : "=l"(ptr)
847           :
848           :);
849     return ptr;
850 }
851
852 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p)
853 {
854     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(0), "r"(p) :);
855 }
856
857 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p)
858 {
859     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(1), "r"(p) :);
860 }
861
862 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p)
863 {
864     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(2), "r"(p) :);
865 }
866
867 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p)
868 {
869     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(3), "r"(p) :);
870 }
871
872 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p)
873 {
874     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(4), "r"(p) :);
875 }
876
```

```
877 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p)
878 {
879     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(5), "r"(p) :);
880 }
881
882 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p)
883 {
884     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(6), "r"(p) :);
885 }
886
887 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p)
888 {
889     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(7), "r"(p) :);
890 }
891
892 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p)
893 {
894     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(8), "r"(p) :);
895 }
896
897 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p)
898 {
899     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(9), "r"(p) :);
900 }
901
902 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p)
903 {
904     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(10), "r"(p) :);
905 }
906
907 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p)
908 {
909     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(11), "r"(p) :);
910 }
911
912 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p)
913 {
914     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(12), "r"(p) :);
915 }
916
917 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p)
918 {
919     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(13), "r"(p) :);
920 }
921
922 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p)
923 {
924     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(14), "r"(p) :);
925 }
926
927 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p)
928 {
929     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(15), "r"(p) :);
930 }
931
932 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p)
933 {
934     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(16), "r"(p) :);
935 }
936
937 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p)
938 {
939     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(17), "r"(p) :);
940 }
941
942 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p)
943 {
```

```
944     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(18), "r"(p) :);
945 }
946
947 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p)
948 {
949     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(19), "r"(p) :);
950 }
951
952 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p)
953 {
954     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(20), "r"(p) :);
955 }
956
957 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p)
958 {
959     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(21), "r"(p) :);
960 }
961
962 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p)
963 {
964     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(22), "r"(p) :);
965 }
966
967 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p)
968 {
969     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(23), "r"(p) :);
970 }
971
972 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p)
973 {
974     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(24), "r"(p) :);
975 }
976
977 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p)
978 {
979     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(25), "r"(p) :);
980 }
981
982 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p)
983 {
984     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(26), "r"(p) :);
985 }
986
987 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p)
988 {
989     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(27), "r"(p) :);
990 }
991
992 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p)
993 {
994     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(28), "r"(p) :);
995 }
996
997 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p)
998 {
999     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(29), "r"(p) :);
1000 }
1001
1002 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p)
1003 {
1004     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(30), "r"(p) :);
1005 }
1006
1007 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p)
1008 {
1009     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(31), "r"(p) :);
1010 }
```

```
1011
1012 static __forceinline__ __device__ unsigned int optixGetPayload_0()
1013 {
1014     unsigned int result;
1015     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(0) :);
1016     return result;
1017 }
1018
1019 static __forceinline__ __device__ unsigned int optixGetPayload_1()
1020 {
1021     unsigned int result;
1022     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(1) :);
1023     return result;
1024 }
1025
1026 static __forceinline__ __device__ unsigned int optixGetPayload_2()
1027 {
1028     unsigned int result;
1029     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(2) :);
1030     return result;
1031 }
1032
1033 static __forceinline__ __device__ unsigned int optixGetPayload_3()
1034 {
1035     unsigned int result;
1036     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(3) :);
1037     return result;
1038 }
1039
1040 static __forceinline__ __device__ unsigned int optixGetPayload_4()
1041 {
1042     unsigned int result;
1043     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(4) :);
1044     return result;
1045 }
1046
1047 static __forceinline__ __device__ unsigned int optixGetPayload_5()
1048 {
1049     unsigned int result;
1050     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(5) :);
1051     return result;
1052 }
1053
1054 static __forceinline__ __device__ unsigned int optixGetPayload_6()
1055 {
1056     unsigned int result;
1057     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(6) :);
1058     return result;
1059 }
1060
1061 static __forceinline__ __device__ unsigned int optixGetPayload_7()
1062 {
1063     unsigned int result;
1064     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(7) :);
1065     return result;
1066 }
1067
1068 static __forceinline__ __device__ unsigned int optixGetPayload_8()
1069 {
1070     unsigned int result;
1071     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(8) :);
1072     return result;
1073 }
1074
1075 static __forceinline__ __device__ unsigned int optixGetPayload_9()
1076 {
1077     unsigned int result;
```

```
1078     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(9) :);
1079     return result;
1080 }
1081
1082 static __forceinline__ __device__ unsigned int optixGetPayload_10()
1083 {
1084     unsigned int result;
1085     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(10) :);
1086     return result;
1087 }
1088
1089 static __forceinline__ __device__ unsigned int optixGetPayload_11()
1090 {
1091     unsigned int result;
1092     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(11) :);
1093     return result;
1094 }
1095
1096 static __forceinline__ __device__ unsigned int optixGetPayload_12()
1097 {
1098     unsigned int result;
1099     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(12) :);
1100     return result;
1101 }
1102
1103 static __forceinline__ __device__ unsigned int optixGetPayload_13()
1104 {
1105     unsigned int result;
1106     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(13) :);
1107     return result;
1108 }
1109
1110 static __forceinline__ __device__ unsigned int optixGetPayload_14()
1111 {
1112     unsigned int result;
1113     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(14) :);
1114     return result;
1115 }
1116
1117 static __forceinline__ __device__ unsigned int optixGetPayload_15()
1118 {
1119     unsigned int result;
1120     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(15) :);
1121     return result;
1122 }
1123
1124 static __forceinline__ __device__ unsigned int optixGetPayload_16()
1125 {
1126     unsigned int result;
1127     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(16) :);
1128     return result;
1129 }
1130
1131 static __forceinline__ __device__ unsigned int optixGetPayload_17()
1132 {
1133     unsigned int result;
1134     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(17) :);
1135     return result;
1136 }
1137
1138 static __forceinline__ __device__ unsigned int optixGetPayload_18()
1139 {
1140     unsigned int result;
1141     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(18) :);
1142     return result;
1143 }
1144
```

```
1145 static __forceinline__ __device__ unsigned int optixGetPayload_19()
1146 {
1147     unsigned int result;
1148     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(19) :);
1149     return result;
1150 }
1151
1152 static __forceinline__ __device__ unsigned int optixGetPayload_20()
1153 {
1154     unsigned int result;
1155     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(20) :);
1156     return result;
1157 }
1158
1159 static __forceinline__ __device__ unsigned int optixGetPayload_21()
1160 {
1161     unsigned int result;
1162     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(21) :);
1163     return result;
1164 }
1165
1166 static __forceinline__ __device__ unsigned int optixGetPayload_22()
1167 {
1168     unsigned int result;
1169     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(22) :);
1170     return result;
1171 }
1172
1173 static __forceinline__ __device__ unsigned int optixGetPayload_23()
1174 {
1175     unsigned int result;
1176     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(23) :);
1177     return result;
1178 }
1179
1180 static __forceinline__ __device__ unsigned int optixGetPayload_24()
1181 {
1182     unsigned int result;
1183     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(24) :);
1184     return result;
1185 }
1186
1187 static __forceinline__ __device__ unsigned int optixGetPayload_25()
1188 {
1189     unsigned int result;
1190     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(25) :);
1191     return result;
1192 }
1193
1194 static __forceinline__ __device__ unsigned int optixGetPayload_26()
1195 {
1196     unsigned int result;
1197     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(26) :);
1198     return result;
1199 }
1200
1201 static __forceinline__ __device__ unsigned int optixGetPayload_27()
1202 {
1203     unsigned int result;
1204     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(27) :);
1205     return result;
1206 }
1207
1208 static __forceinline__ __device__ unsigned int optixGetPayload_28()
1209 {
1210     unsigned int result;
1211     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(28) :);
```

```
1212     return result;
1213 }
1214
1215 static __forceinline__ __device__ unsigned int optixGetPayload_29()
1216 {
1217     unsigned int result;
1218     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(29) :);
1219     return result;
1220 }
1221
1222 static __forceinline__ __device__ unsigned int optixGetPayload_30()
1223 {
1224     unsigned int result;
1225     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(30) :);
1226     return result;
1227 }
1228
1229 static __forceinline__ __device__ unsigned int optixGetPayload_31()
1230 {
1231     unsigned int result;
1232     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(31) :);
1233     return result;
1234 }
1235
1236 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int types)
1237 {
1238     asm volatile("call _optix_set_payload_types, (%0);" : : "r"(types) :);
1239 }
1240
1241 static __forceinline__ __device__ unsigned int optixUndefinedValue()
1242 {
1243     unsigned int u0;
1244     asm("call (%0), _optix_undef_value, ();" : "=r"(u0) :);
1245     return u0;
1246 }
1247
1248 static __forceinline__ __device__ float3 optixGetWorldRayOrigin()
1249 {
1250     float f0, f1, f2;
1251     asm("call (%0), _optix_get_world_ray_origin_x, ();" : "=f"(f0) :);
1252     asm("call (%0), _optix_get_world_ray_origin_y, ();" : "=f"(f1) :);
1253     asm("call (%0), _optix_get_world_ray_origin_z, ();" : "=f"(f2) :);
1254     return make_float3(f0, f1, f2);
1255 }
1256
1257 static __forceinline__ __device__ float3 optixGetWorldRayDirection()
1258 {
1259     float f0, f1, f2;
1260     asm("call (%0), _optix_get_world_ray_direction_x, ();" : "=f"(f0) :);
1261     asm("call (%0), _optix_get_world_ray_direction_y, ();" : "=f"(f1) :);
1262     asm("call (%0), _optix_get_world_ray_direction_z, ();" : "=f"(f2) :);
1263     return make_float3(f0, f1, f2);
1264 }
1265
1266 static __forceinline__ __device__ float3 optixGetObjectRayOrigin()
1267 {
1268     float f0, f1, f2;
1269     asm("call (%0), _optix_get_object_ray_origin_x, ();" : "=f"(f0) :);
1270     asm("call (%0), _optix_get_object_ray_origin_y, ();" : "=f"(f1) :);
1271     asm("call (%0), _optix_get_object_ray_origin_z, ();" : "=f"(f2) :);
1272     return make_float3(f0, f1, f2);
1273 }
1274
1275 static __forceinline__ __device__ float3 optixGetObjectRayDirection()
1276 {
1277     float f0, f1, f2;
1278     asm("call (%0), _optix_get_object_ray_direction_x, ();" : "=f"(f0) :);
```

```
1279     asm("call (%0), _optix_get_object_ray_direction_y, ();" : "=f"(f1) :);
1280     asm("call (%0), _optix_get_object_ray_direction_z, ();" : "=f"(f2) :);
1281     return make_float3(f0, f1, f2);
1282 }
1283
1284 static __forceinline__ __device__ float optixGetRayTmin()
1285 {
1286     float f0;
1287     asm("call (%0), _optix_get_ray_tmin, ();" : "=f"(f0) :);
1288     return f0;
1289 }
1290
1291 static __forceinline__ __device__ float optixGetRayTmax()
1292 {
1293     float f0;
1294     asm("call (%0), _optix_get_ray_tmax, ();" : "=f"(f0) :);
1295     return f0;
1296 }
1297
1298 static __forceinline__ __device__ float optixGetRayTime()
1299 {
1300     float f0;
1301     asm("call (%0), _optix_get_ray_time, ();" : "=f"(f0) :);
1302     return f0;
1303 }
1304
1305 static __forceinline__ __device__ unsigned int optixGetRayFlags()
1306 {
1307     unsigned int u0;
1308     asm("call (%0), _optix_get_ray_flags, ();" : "=r"(u0) :);
1309     return u0;
1310 }
1311
1312 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask()
1313 {
1314     unsigned int u0;
1315     asm("call (%0), _optix_get_ray_visibility_mask, ();" : "=r"(u0) :);
1316     return u0;
1317 }
1318
1319 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias,
1320                                                                                 unsigned int
instIdx)
1321 {
1322     unsigned long long handle;
1323     asm("call (%0), _optix_get_instance_traversable_from_ias, (%1, %2);"
1324         : "=l"(handle) : "l"(ias), "r"(instIdx));
1325     return (OptixTraversableHandle)handle;
1326 }
1327
1328
1329 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas,
1330                                                                    unsigned int           primIdx,
1331                                                                    unsigned int           sbtGASIndex,
1332                                                                    float                  time,
1333                                                                    float3                 data[3])
1334 {
1335     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_triangle_vertex_data, "
1336         "(%9, %10, %11, %12);"
1337         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
1338           "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1339         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1340         :);
1341 }
1342
1343 static __forceinline__ __device__ void optixGetMicroTriangleVertexData(float3 data[3])
```

```
1344 {
1345     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_microtriangle_vertex_data, "
1346         "();"
1347         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
1348           "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1349         :);
1350 }
1351 static __forceinline__ __device__ void optixGetMicroTriangleBarycentricsData(float2 data[3])
1352 {
1353   asm("call (%0, %1, %2, %3, %4, %5), _optix_get_microtriangle_barycentrics_data, "
1354       "();"
1355       : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[1].x), "=f"(data[1].y), "=f"(data[2].x),
"=f"(data[2].y)
1356       :);
1357 }
1358
1359 static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
1360                                                                       unsigned int           primIdx,
1361                                                                     unsigned int           sbtGASIndex,
1362                                                                       float                  time,
1363                                                                       float4                 data[2])
1364 {
1365     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7), _optix_get_linear_curve_vertex_data, "
1366         "(%8, %9, %10, %11);"
1367         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1368           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
1369         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1370         :);
1371 }
1372
1373 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(OptixTraversableHandle gas,
1374                                                                           unsigned int         primIdx,
1375                                                                         unsigned int         sbtGASIndex,
1376                                                                           float                time,
1377                                                                           float4               data[3])
1378 {
1379     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11),
_optix_get_quadratic_bspline_vertex_data, "
1380         "(%12, %13, %14, %15);"
1381         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1382           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1383           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w)
1384         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1385         :);
1386 }
1387
1388 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,
1389                                                                       unsigned int         primIdx,
1390                                                                     unsigned int         sbtGASIndex,
1391                                                                       float                time,
1392                                                                       float4               data[4])
1393 {
1394     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1395         "_optix_get_cubic_bspline_vertex_data, "
1396         "(%16, %17, %18, %19);"
1397         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1398           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1399           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
1400           "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
1401         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1402         :);
1403 }
1404
1405 static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas,
1406                                                                       unsigned int           primIdx,
1407                                                                     unsigned int           sbtGASIndex,
1408                                                                       float                  time,
```

```
1409                                                                    float4                 data[4])
1410 {
1411     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1412         "_optix_get_catmullrom_vertex_data, "
1413         "(%16, %17, %18, %19);"
1414         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1415           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1416           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1417         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1418         :);
1419 }
1420
1421 static __forceinline__ __device__ void optixGetCubicBezierVertexData(OptixTraversableHandle gas,
1422                                                                       unsigned int           primIdx,
1423                                                                     unsigned int           sbtGASIndex,
1424                                                                       float                  time,
1425                                                                       float4                 data[4])
1426 {
1427     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1428         "_optix_get_cubic_bezier_vertex_data, "
1429         "(%16, %17, %18, %19);"
1430         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1431           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1432           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1433         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1434         :);
1435 }
1436
1437 static __forceinline__ __device__ void optixGetRibbonVertexData(OptixTraversableHandle gas,
1438                                                                 unsigned int           primIdx,
1439                                                                 unsigned int           sbtGASIndex,
1440                                                                 float                  time,
1441                                                                 float4                 data[3])
1442 {
1443     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11), _optix_get_ribbon_vertex_data, "
1444         "(%12, %13, %14, %15);"
1445         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
"=f"(data[1].y),
1446           "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
"=f"(data[2].w)
1447         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1448         :);
1449 }
1450
1451 static __forceinline__ __device__ float3 optixGetRibbonNormal(OptixTraversableHandle gas,
1452                                                               unsigned int           primIdx,
1453                                                               unsigned int           sbtGASIndex,
1454                                                               float                  time,
1455                                                               float2                 ribbonParameters)
1456 {
1457     float3 normal;
1458     asm("call (%0, %1, %2), _optix_get_ribbon_normal, "
1459         "(%3, %4, %5, %6, %7, %8);"
1460         : "=f"(normal.x), "=f"(normal.y), "=f"(normal.z)
1461         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time),
1462           "f"(ribbonParameters.x), "f"(ribbonParameters.y)
1463         :);
1464     return normal;
1465 }
1466
1467 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas,
1468                                                           unsigned int           primIdx,
1469                                                           unsigned int           sbtGASIndex,
1470                                                           float                  time,
1471                                                           float4                 data[1])
```

```
1472 {
1473     asm("call (%0, %1, %2, %3), "
1474         "_optix_get_sphere_data, "
1475         "(%4, %5, %6, %7);"
1476         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
1477         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1478         :);
1479 }
1480
1481 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle()
1482 {
1483     unsigned long long handle;
1484     asm("call (%0), _optix_get_gas_traversable_handle, ();" : "=l"(handle) :);
1485     return (OptixTraversableHandle)handle;
1486 }
1487
1488 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle handle)
1489 {
1490     float f0;
1491     asm("call (%0), _optix_get_gas_motion_time_begin, (%1);" : "=f"(f0) : "l"(handle) :);
1492     return f0;
1493 }
1494
1495 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle handle)
1496 {
1497     float f0;
1498     asm("call (%0), _optix_get_gas_motion_time_end, (%1);" : "=f"(f0) : "l"(handle) :);
1499     return f0;
1500 }
1501
1502 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle handle)
1503 {
1504     unsigned int u0;
1505     asm("call (%0), _optix_get_gas_motion_step_count, (%1);" : "=r"(u0) : "l"(handle) :);
1506     return u0;
1507 }
1508
1509 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12])
1510 {
1511     if(optixGetTransformListSize() == 0)
1512     {
1513         m[0]  = 1.0f;
1514         m[1]  = 0.0f;
1515         m[2]  = 0.0f;
1516         m[3]  = 0.0f;
1517         m[4]  = 0.0f;
1518         m[5]  = 1.0f;
1519         m[6]  = 0.0f;
1520         m[7]  = 0.0f;
1521         m[8]  = 0.0f;
1522         m[9]  = 0.0f;
1523         m[10] = 1.0f;
1524         m[11] = 0.0f;
1525         return;
1526     }
1527
1528     float4 m0, m1, m2;
1529     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
1530     m[0]  = m0.x;
1531     m[1]  = m0.y;
1532     m[2]  = m0.z;
1533     m[3]  = m0.w;
1534     m[4]  = m1.x;
1535     m[5]  = m1.y;
1536     m[6]  = m1.z;
1537     m[7]  = m1.w;
1538     m[8]  = m2.x;
```

```
1539     m[9]  = m2.y;
1540     m[10] = m2.z;
1541     m[11] = m2.w;
1542 }
1543
1544 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12])
1545 {
1546     if(optixGetTransformListSize() == 0)
1547     {
1548         m[0]  = 1.0f;
1549         m[1]  = 0.0f;
1550         m[2]  = 0.0f;
1551         m[3]  = 0.0f;
1552         m[4]  = 0.0f;
1553         m[5]  = 1.0f;
1554         m[6]  = 0.0f;
1555         m[7]  = 0.0f;
1556         m[8]  = 0.0f;
1557         m[9]  = 0.0f;
1558         m[10] = 1.0f;
1559         m[11] = 0.0f;
1560         return;
1561     }
1562
1563     float4 m0, m1, m2;
1564     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);
1565     m[0]  = m0.x;
1566     m[1]  = m0.y;
1567     m[2]  = m0.z;
1568     m[3]  = m0.w;
1569     m[4]  = m1.x;
1570     m[5]  = m1.y;
1571     m[6]  = m1.z;
1572     m[7]  = m1.w;
1573     m[8]  = m2.x;
1574     m[9]  = m2.y;
1575     m[10] = m2.z;
1576     m[11] = m2.w;
1577 }
1578
1579 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point)
1580 {
1581     if(optixGetTransformListSize() == 0)
1582         return point;
1583
1584     float4 m0, m1, m2;
1585     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
1586     return optix_impl::optixTransformPoint(m0, m1, m2, point);
1587 }
1588
1589 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec)
1590 {
1591     if(optixGetTransformListSize() == 0)
1592         return vec;
1593
1594     float4 m0, m1, m2;
1595     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);
1596     return optix_impl::optixTransformVector(m0, m1, m2, vec);
1597 }
1598
1599 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal)
1600 {
1601     if(optixGetTransformListSize() == 0)
1602         return normal;
1603
1604     float4 m0, m1, m2;
1605     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);  // inverse of
```

```
optixGetWorldToObjectTransformMatrix()
1606     return optix_impl::optixTransformNormal(m0, m1, m2, normal);
1607 }
1608
1609 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point)
1610 {
1611     if(optixGetTransformListSize() == 0)
1612         return point;
1613
1614     float4 m0, m1, m2;
1615     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);
1616     return optix_impl::optixTransformPoint(m0, m1, m2, point);
1617 }
1618
1619 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec)
1620 {
1621     if(optixGetTransformListSize() == 0)
1622         return vec;
1623
1624     float4 m0, m1, m2;
1625     optix_impl::optixGetObjectToWorldTransformMatrix(m0, m1, m2);
1626     return optix_impl::optixTransformVector(m0, m1, m2, vec);
1627 }
1628
1629 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal)
1630 {
1631     if(optixGetTransformListSize() == 0)
1632         return normal;
1633
1634     float4 m0, m1, m2;
1635     optix_impl::optixGetWorldToObjectTransformMatrix(m0, m1, m2);  // inverse of
optixGetObjectToWorldTransformMatrix()
1636     return optix_impl::optixTransformNormal(m0, m1, m2, normal);
1637 }
1638
1639 static __forceinline__ __device__ unsigned int optixGetTransformListSize()
1640 {
1641     unsigned int u0;
1642     asm("call (%0), _optix_get_transform_list_size, ();" : "=r"(u0) :);
1643     return u0;
1644 }
1645
1646 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int index)
1647 {
1648     unsigned long long u0;
1649     asm("call (%0), _optix_get_transform_list_handle, (%1);" : "=l"(u0) : "r"(index) :);
1650     return u0;
1651 }
1652
1653 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle)
1654 {
1655     int i0;
1656     asm("call (%0), _optix_get_transform_type_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
1657     return (OptixTransformType)i0;
1658 }
1659
1660 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle)
1661 {
1662     unsigned long long ptr;
1663     asm("call (%0), _optix_get_static_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
1664     return (const OptixStaticTransform*)ptr;
1665 }
1666
1667 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle)
```

```
1668 {
1669     unsigned long long ptr;
1670     asm("call (%0), _optix_get_srt_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
1671     return (const OptixSRTMotionTransform*)ptr;
1672 }
1673
1674 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle)
1675 {
1676     unsigned long long ptr;
1677     asm("call (%0), _optix_get_matrix_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
1678     return (const OptixMatrixMotionTransform*)ptr;
1679 }
1680
1681 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle)
1682 {
1683     int i0;
1684     asm("call (%0), _optix_get_instance_id_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
1685     return i0;
1686 }
1687
1688 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle(OptixTraversableHandle handle)
1689 {
1690     unsigned long long i0;
1691     asm("call (%0), _optix_get_instance_child_from_handle, (%1);" : "=l"(i0) : "l"(handle) :);
1692     return (OptixTraversableHandle)i0;
1693 }
1694
1695 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle)
1696 {
1697     unsigned long long ptr;
1698     asm("call (%0), _optix_get_instance_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
1699     return (const float4*)ptr;
1700 }
1701
1702 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle)
1703 {
1704     unsigned long long ptr;
1705     asm("call (%0), _optix_get_instance_inverse_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle)
:);
1706     return (const float4*)ptr;
1707 }
1708
1709 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind)
1710 {
1711     int ret;
1712     asm volatile(
1713         "call (%0), _optix_report_intersection_0"
1714         ", (%1, %2);"
1715         : "=r"(ret)
1716         : "f"(hitT), "r"(hitKind)
1717         :);
1718     return ret;
1719 }
1720
1721 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0)
1722 {
1723     int ret;
1724     asm volatile(
1725         "call (%0), _optix_report_intersection_1"
1726         ", (%1, %2, %3);"
1727         : "=r"(ret)
```

```
1728         : "f"(hitT), "r"(hitKind), "r"(a0)
1729         :);
1730     return ret;
1731 }
1732
1733 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1)
1734 {
1735     int ret;
1736     asm volatile(
1737         "call (%0), _optix_report_intersection_2"
1738         ", (%1, %2, %3, %4);"
1739         : "=r"(ret)
1740         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1)
1741         :);
1742     return ret;
1743 }
1744
1745 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1, unsigned int a2)
1746 {
1747     int ret;
1748     asm volatile(
1749         "call (%0), _optix_report_intersection_3"
1750         ", (%1, %2, %3, %4, %5);"
1751         : "=r"(ret)
1752         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2)
1753         :);
1754     return ret;
1755 }
1756
1757 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1758                                                                unsigned int hitKind,
1759                                                                unsigned int a0,
1760                                                                unsigned int a1,
1761                                                                unsigned int a2,
1762                                                                unsigned int a3)
1763 {
1764     int ret;
1765     asm volatile(
1766         "call (%0), _optix_report_intersection_4"
1767         ", (%1, %2, %3, %4, %5, %6);"
1768         : "=r"(ret)
1769         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3)
1770         :);
1771     return ret;
1772 }
1773
1774 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1775                                                                unsigned int hitKind,
1776                                                                unsigned int a0,
1777                                                                unsigned int a1,
1778                                                                unsigned int a2,
1779                                                                unsigned int a3,
1780                                                                unsigned int a4)
1781 {
1782     int ret;
1783     asm volatile(
1784         "call (%0), _optix_report_intersection_5"
1785         ", (%1, %2, %3, %4, %5, %6, %7);"
1786         : "=r"(ret)
1787         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4)
1788         :);
1789     return ret;
1790 }
1791
1792 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
```

```
1793                                                                unsigned int hitKind,
1794                                                                unsigned int a0,
1795                                                                unsigned int a1,
1796                                                                unsigned int a2,
1797                                                                unsigned int a3,
1798                                                                unsigned int a4,
1799                                                                unsigned int a5)
1800 {
1801     int ret;
1802     asm volatile(
1803         "call (%0), _optix_report_intersection_6"
1804         ", (%1, %2, %3, %4, %5, %6, %7, %8);"
1805         : "=r"(ret)
1806         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5)
1807         :);
1808     return ret;
1809 }
1810
1811 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1812                                                                unsigned int hitKind,
1813                                                                unsigned int a0,
1814                                                                unsigned int a1,
1815                                                                unsigned int a2,
1816                                                                unsigned int a3,
1817                                                                unsigned int a4,
1818                                                                unsigned int a5,
1819                                                                unsigned int a6)
1820 {
1821     int ret;
1822     asm volatile(
1823         "call (%0), _optix_report_intersection_7"
1824         ", (%1, %2, %3, %4, %5, %6, %7, %8, %9);"
1825         : "=r"(ret)
1826         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6)
1827         :);
1828     return ret;
1829 }
1830
1831 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1832                                                                unsigned int hitKind,
1833                                                                unsigned int a0,
1834                                                                unsigned int a1,
1835                                                                unsigned int a2,
1836                                                                unsigned int a3,
1837                                                                unsigned int a4,
1838                                                                unsigned int a5,
1839                                                                unsigned int a6,
1840                                                                unsigned int a7)
1841 {
1842     int ret;
1843     asm volatile(
1844         "call (%0), _optix_report_intersection_8"
1845         ", (%1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
1846         : "=r"(ret)
1847        : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6), "r"(a7)
1848         :);
1849     return ret;
1850 }
1851
1852 #define OPTIX_DEFINE_optixGetAttribute_BODY(which)
\
1853 unsigned int ret;
\
1854 asm("call (%0), _optix_get_attribute_" #which ", ();" : "=r"(ret) :);
\
1855     return ret;
1856
```

```
1857 static __forceinline__ __device__ unsigned int optixGetAttribute_0()
1858 {
1859     OPTIX_DEFINE_optixGetAttribute_BODY(0);
1860 }
1861
1862 static __forceinline__ __device__ unsigned int optixGetAttribute_1()
1863 {
1864     OPTIX_DEFINE_optixGetAttribute_BODY(1);
1865 }
1866
1867 static __forceinline__ __device__ unsigned int optixGetAttribute_2()
1868 {
1869     OPTIX_DEFINE_optixGetAttribute_BODY(2);
1870 }
1871
1872 static __forceinline__ __device__ unsigned int optixGetAttribute_3()
1873 {
1874     OPTIX_DEFINE_optixGetAttribute_BODY(3);
1875 }
1876
1877 static __forceinline__ __device__ unsigned int optixGetAttribute_4()
1878 {
1879     OPTIX_DEFINE_optixGetAttribute_BODY(4);
1880 }
1881
1882 static __forceinline__ __device__ unsigned int optixGetAttribute_5()
1883 {
1884     OPTIX_DEFINE_optixGetAttribute_BODY(5);
1885 }
1886
1887 static __forceinline__ __device__ unsigned int optixGetAttribute_6()
1888 {
1889     OPTIX_DEFINE_optixGetAttribute_BODY(6);
1890 }
1891
1892 static __forceinline__ __device__ unsigned int optixGetAttribute_7()
1893 {
1894     OPTIX_DEFINE_optixGetAttribute_BODY(7);
1895 }
1896
1897 #undef OPTIX_DEFINE_optixGetAttribute_BODY
1898
1899 static __forceinline__ __device__ void optixTerminateRay()
1900 {
1901     asm volatile("call _optix_terminate_ray, ();");
1902 }
1903
1904 static __forceinline__ __device__ void optixIgnoreIntersection()
1905 {
1906     asm volatile("call _optix_ignore_intersection, ();");
1907 }
1908
1909 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex()
1910 {
1911     unsigned int u0;
1912     asm("call (%0), _optix_read_primitive_idx, ();" : "=r"(u0) :);
1913     return u0;
1914 }
1915
1916 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex()
1917 {
1918     unsigned int u0;
1919     asm("call (%0), _optix_read_sbt_gas_idx, ();" : "=r"(u0) :);
1920     return u0;
1921 }
1922
1923 static __forceinline__ __device__ unsigned int optixGetInstanceId()
```

```
1924 {
1925     unsigned int u0;
1926     asm("call (%0), _optix_read_instance_id, ();" : "=r"(u0) :);
1927     return u0;
1928 }
1929
1930 static __forceinline__ __device__ unsigned int optixGetInstanceIndex()
1931 {
1932     unsigned int u0;
1933     asm("call (%0), _optix_read_instance_idx, ();" : "=r"(u0) :);
1934     return u0;
1935 }
1936
1937 static __forceinline__ __device__ unsigned int optixGetHitKind()
1938 {
1939     unsigned int u0;
1940     asm("call (%0), _optix_get_hit_kind, ();" : "=r"(u0) :);
1941     return u0;
1942 }
1943
1944 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind)
1945 {
1946     unsigned int u0;
1947     asm("call (%0), _optix_get_primitive_type_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));
1948     return (OptixPrimitiveType)u0;
1949 }
1950
1951 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind)
1952 {
1953     unsigned int u0;
1954     asm("call (%0), _optix_get_backface_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));
1955     return (u0 == 0x1);
1956 }
1957
1958 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind)
1959 {
1960     return !optixIsBackFaceHit(hitKind);
1961 }
1962
1963
1964 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType()
1965 {
1966     return optixGetPrimitiveType(optixGetHitKind());
1967 }
1968
1969 static __forceinline__ __device__ bool optixIsBackFaceHit()
1970 {
1971     return optixIsBackFaceHit(optixGetHitKind());
1972 }
1973
1974 static __forceinline__ __device__ bool optixIsFrontFaceHit()
1975 {
1976     return optixIsFrontFaceHit(optixGetHitKind());
1977 }
1978
1979 static __forceinline__ __device__ bool optixIsTriangleHit()
1980 {
1981     return optixIsTriangleFrontFaceHit() || optixIsTriangleBackFaceHit();
1982 }
1983
1984 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit()
1985 {
1986     return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE;
1987 }
1988
1989 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit()
1990 {
```

```
1991     return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_BACK_FACE;
1992 }
1993
1994 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit()
1995 {
1996     return optixGetPrimitiveType(optixGetHitKind()) ==
OPTIX_PRIMITIVE_TYPE_DISPLACED_MICROMESH_TRIANGLE;
1997 }
1998
1999 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleFrontFaceHit()
2000 {
2001     return optixIsDisplacedMicromeshTriangleHit() && optixIsFrontFaceHit();
2002 }
2003
2004 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleBackFaceHit()
2005 {
2006     return optixIsDisplacedMicromeshTriangleHit() && optixIsBackFaceHit();
2007 }
2008
2009 static __forceinline__ __device__ float optixGetCurveParameter()
2010 {
2011     float f0;
2012     asm("call (%0), _optix_get_curve_parameter, ();" : "=f"(f0) :);
2013     return f0;
2014 }
2015
2016 static __forceinline__ __device__ float2 optixGetRibbonParameters()
2017 {
2018     float f0, f1;
2019     asm("call (%0, %1), _optix_get_ribbon_parameters, ();" : "=f"(f0), "=f"(f1) :);
2020     return make_float2(f0, f1);
2021 }
2022
2023 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics()
2024 {
2025     float f0, f1;
2026     asm("call (%0, %1), _optix_get_triangle_barycentrics, ();" : "=f"(f0), "=f"(f1) :);
2027     return make_float2(f0, f1);
2028 }
2029
2030 static __forceinline__ __device__ uint3 optixGetLaunchIndex()
2031 {
2032     unsigned int u0, u1, u2;
2033     asm("call (%0), _optix_get_launch_index_x, ();" : "=r"(u0) :);
2034     asm("call (%0), _optix_get_launch_index_y, ();" : "=r"(u1) :);
2035     asm("call (%0), _optix_get_launch_index_z, ();" : "=r"(u2) :);
2036     return make_uint3(u0, u1, u2);
2037 }
2038
2039 static __forceinline__ __device__ uint3 optixGetLaunchDimensions()
2040 {
2041     unsigned int u0, u1, u2;
2042     asm("call (%0), _optix_get_launch_dimension_x, ();" : "=r"(u0) :);
2043     asm("call (%0), _optix_get_launch_dimension_y, ();" : "=r"(u1) :);
2044     asm("call (%0), _optix_get_launch_dimension_z, ();" : "=r"(u2) :);
2045     return make_uint3(u0, u1, u2);
2046 }
2047
2048 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer()
2049 {
2050     unsigned long long ptr;
2051     asm("call (%0), _optix_get_sbt_data_ptr_64, ();" : "=l"(ptr) :);
2052     return (CUdeviceptr)ptr;
2053 }
2054
2055 static __forceinline__ __device__ void optixThrowException(int exceptionCode)
2056 {
```

```
2057     asm volatile(
2058         "call _optix_throw_exception_0, (%0);"
2059         : /* no return value */
2060         : "r"(exceptionCode)
2061         :);
2062 }
2063
2064 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0)
2065 {
2066     asm volatile(
2067         "call _optix_throw_exception_1, (%0, %1);"
2068         : /* no return value */
2069         : "r"(exceptionCode), "r"(exceptionDetail0)
2070         :);
2071 }
2072
2073 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1)
2074 {
2075     asm volatile(
2076         "call _optix_throw_exception_2, (%0, %1, %2);"
2077         : /* no return value */
2078         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1)
2079         :);
2080 }
2081
2082 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
2083 {
2084     asm volatile(
2085         "call _optix_throw_exception_3, (%0, %1, %2, %3);"
2086         : /* no return value */
2087         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2)
2088         :);
2089 }
2090
2091 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3)
2092 {
2093     asm volatile(
2094         "call _optix_throw_exception_4, (%0, %1, %2, %3, %4);"
2095         : /* no return value */
2096         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3)
2097         :);
2098 }
2099
2100 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3, unsigned int exceptionDetail4)
2101 {
2102     asm volatile(
2103         "call _optix_throw_exception_5, (%0, %1, %2, %3, %4, %5);"
2104         : /* no return value */
2105         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4)
2106         :);
2107 }
2108
2109 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)
2110 {
2111     asm volatile(
2112         "call _optix_throw_exception_6, (%0, %1, %2, %3, %4, %5, %6);"
```

```
2113         : /* no return value */
2114         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5)
2115         :);
2116 }
2117
2118 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)
2119 {
2120     asm volatile(
2121         "call _optix_throw_exception_7, (%0, %1, %2, %3, %4, %5, %6, %7);"
2122         : /* no return value */
2123         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6)
2124         :);
2125 }
2126
2127 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)
2128 {
2129     asm volatile(
2130         "call _optix_throw_exception_8, (%0, %1, %2, %3, %4, %5, %6, %7, %8);"
2131         : /* no return value */
2132         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6),
"r"(exceptionDetail7)
2133         :);
2134 }
2135
2136 static __forceinline__ __device__ int optixGetExceptionCode( )
2137 {
2138     int s0;
2139     asm("call (%0), _optix_get_exception_code, ();" : "=r"(s0) :);
2140     return s0;
2141 }
2142
2143 #define OPTIX_DEFINE_optixGetExceptionDetail_BODY(which)
\
2144 unsigned int ret;
\
2145 asm("call (%0), _optix_get_exception_detail_" #which ", ();" : "=r"(ret) :);
\
2146     return ret;
2147
2148 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0( )
2149 {
2150     OPTIX_DEFINE_optixGetExceptionDetail_BODY(0);
2151 }
2152
2153 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1( )
2154 {
2155     OPTIX_DEFINE_optixGetExceptionDetail_BODY(1);
2156 }
2157
2158 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2( )
2159 {
2160     OPTIX_DEFINE_optixGetExceptionDetail_BODY(2);
2161 }
2162
2163 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3( )
2164 {
2165     OPTIX_DEFINE_optixGetExceptionDetail_BODY(3);
2166 }
```

```
2167
2168 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4()
2169 {
2170     OPTIX_DEFINE_optixGetExceptionDetail_BODY(4);
2171 }
2172
2173 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5()
2174 {
2175     OPTIX_DEFINE_optixGetExceptionDetail_BODY(5);
2176 }
2177
2178 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6()
2179 {
2180     OPTIX_DEFINE_optixGetExceptionDetail_BODY(6);
2181 }
2182
2183 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7()
2184 {
2185     OPTIX_DEFINE_optixGetExceptionDetail_BODY(7);
2186 }
2187
2188 #undef OPTIX_DEFINE_optixGetExceptionDetail_BODY
2189
2190
2191 static __forceinline__ __device__ char* optixGetExceptionLineInfo()
2192 {
2193     unsigned long long ptr;
2194     asm("call (%0), _optix_get_exception_line_info, ();" : "=l"(ptr) :);
2195     return (char*)ptr;
2196 }
2197
2198 template <typename ReturnT, typename... ArgTypes>
2199 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args)
2200 {
2201     unsigned long long func;
2202     asm("call (%0), _optix_call_direct_callable,(%1);" : "=l"(func) : "r"(sbtIndex) :);
2203     using funcT = ReturnT (*)(ArgTypes...);
2204     funcT call  = (funcT)(func);
2205     return call(args...);
2206 }
2207
2208 template <typename ReturnT, typename... ArgTypes>
2209 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes... args)
2210 {
2211     unsigned long long func;
2212     asm("call (%0), _optix_call_continuation_callable,(%1);" : "=l"(func) : "r"(sbtIndex) :);
2213     using funcT = ReturnT (*)(ArgTypes...);
2214     funcT call  = (funcT)(func);
2215     return call(args...);
2216 }
2217
2218 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int
texInfo, float x, float y, unsigned int* singleMipLevel)
2219 {
2220     uint4              result;
2221     unsigned long long resultPtr         = reinterpret_cast<unsigned long long>(&result);
2222     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2223     // Cast float args to integers, because the intrinics take .b32 arguments when compiled to PTX.
2224     asm volatile(
2225         "call _optix_tex_footprint_2d_v2"
2226         ", (%0, %1, %2, %3, %4, %5);"
2227         :
2228         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2229           "l"(singleMipLevelPtr), "l"(resultPtr)
2230         :);
2231     return result;
2232 }
```

```
2233
2234 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
2235                                                                  unsigned int       texInfo,
2236                                                                  float              x,
2237                                                                  float              y,
2238                                                                  float              dPdx_x,
2239                                                                  float              dPdx_y,
2240                                                                  float              dPdy_x,
2241                                                                  float              dPdy_y,
2242                                                                  bool               coarse,
2243                                                                  unsigned int*      singleMipLevel)
2244 {
2245     uint4              result;
2246     unsigned long long resultPtr         = reinterpret_cast<unsigned long long>(&result);
2247     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2248     // Cast float args to integers, because the intrinics take .b32 arguments when compiled to PTX.
2249     asm volatile(
2250         "call _optix_tex_footprint_2d_grad_v2"
2251         ", (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
2252         :
2253         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2254           "r"(__float_as_uint(dPdx_x)), "r"(__float_as_uint(dPdx_y)), "r"(__float_as_uint(dPdy_x)),
2255           "r"(__float_as_uint(dPdy_y)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
"l"(resultPtr)
2256         :);
2257
2258     return result;
2259 }
2260
2261 static __forceinline__ __device__ uint4
2262 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool
coarse, unsigned int* singleMipLevel)
2263 {
2264     uint4              result;
2265     unsigned long long resultPtr         = reinterpret_cast<unsigned long long>(&result);
2266     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2267     // Cast float args to integers, because the intrinics take .b32 arguments when compiled to PTX.
2268     asm volatile(
2269         "call _optix_tex_footprint_2d_lod_v2"
2270         ", (%0, %1, %2, %3, %4, %5, %6, %7);"
2271         :
2272         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2273           "r"(__float_as_uint(level)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
"l"(resultPtr)
2274         :);
2275     return result;
2276 }
2277
2278 #endif // OPTIX_OPTIX_DEVICE_IMPL_H
```

8.3 optix_device_impl_transformations.h File Reference

Namespaces

- namespace optix_impl

Functions

- static __forceinline__ __device__ float4 optix_impl::optixAddFloat4 (const float4 &a, const float4 &b)
- static __forceinline__ __device__ float4 optix_impl::optixMulFloat4 (const float4 &a, float b)
- static __forceinline__ __device__ uint4 optix_impl::optixLdg (unsigned long long addr)
- template<class T >
  static __forceinline__ __device__ T optix_impl::optixLoadReadOnlyAlign16 (const T *ptr)

- static __forceinline__ __device__ float4 optix_impl::optixMultiplyRowMatrix (const float4 vec, const float4 m0, const float4 m1, const float4 m2)
- static __forceinline__ __device__ void optix_impl::optixGetMatrixFromSrt (float4 &m0, float4 &m1, float4 &m2, const OptixSRTData &srt)
- static __forceinline__ __device__ void optix_impl::optixInvertMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ void optix_impl::optixLoadInterpolatedMatrixKey (float4 &m0, float4 &m1, float4 &m2, const float4 *matrix, const float t1)
- static __forceinline__ __device__ void optix_impl::optixLoadInterpolatedSrtKey (float4 &srt0, float4 &srt1, float4 &srt2, float4 &srt3, const float4 *srt, const float t1)
- static __forceinline__ __device__ void optix_impl::optixResolveMotionKey (float &localt, int &key, const OptixMotionOptions &options, const float globalt)
- static __forceinline__ __device__ void optix_impl::optixGetInterpolatedTransformation (float4 &trf0, float4 &trf1, float4 &trf2, const OptixMatrixMotionTransform *transformData, const float time)
- static __forceinline__ __device__ void optix_impl::optixGetInterpolatedTransformation (float4 &trf0, float4 &trf1, float4 &trf2, const OptixSRTMotionTransform *transformData, const float time)
- static __forceinline__ __device__ void optix_impl ::optixGetInterpolatedTransformationFromHandle (float4 &trf0, float4 &trf1, float4 &trf2, const OptixTraversableHandle handle, const float time, const bool objectToWorld)
- static __forceinline__ __device__ void optix_impl::optixGetWorldToObjectTransformMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ void optix_impl::optixGetObjectToWorldTransformMatrix (float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ float3 optix_impl::optixTransformPoint (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &p)
- static __forceinline__ __device__ float3 optix_impl::optixTransformVector (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &v)
- static __forceinline__ __device__ float3 optix_impl::optixTransformNormal (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &n)

8.3.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Device side implementation for transformation helper functions.

8.4 optix_device_impl_transformations.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2023 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_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_DEVICE_IMPL_TRANSFORMATIONS_H
34 #define OPTIX_OPTIX_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     // Debug mode may keep this temporary variable
63     // If T does not enforce 16B alignment, v may not be 16B aligned and storing the loaded data from ptr
fails
64     __align__(16) T v;
65     for(int ofs                    = 0; ofs < sizeof(T); ofs += 16)
66         *(uint4*)((char*)&v + ofs) = optixLdg((unsigned long long)((char*)ptr + ofs));
67     return v;
68 }
69
70 // Multiplies the row vector vec with the 3x4 matrix with rows m0, m1, and m2
71 static __forceinline__ __device__ float4 optixMultiplyRowMatrix(const float4 vec, const float4 m0, const
float4 m1, const float4 m2)
72 {
73     float4 result;
74
75     result.x = vec.x * m0.x + vec.y * m1.x + vec.z * m2.x;
76     result.y = vec.x * m0.y + vec.y * m1.y + vec.z * m2.y;
77     result.z = vec.x * m0.z + vec.y * m1.z + vec.z * m2.z;
78     result.w = vec.x * m0.w + vec.y * m1.w + vec.z * m2.w + vec.w;
79
80     return result;
81 }
82
```

```
83 // Converts the SRT transformation srt into a 3x4 matrix with rows m0, m1, and m2
84 static __forceinline__ __device__ void optixGetMatrixFromSrt(float4& m0, float4& m1, float4& m2, const
OptixSRTData& srt)
85 {
86     const float4 q = {srt.qx, srt.qy, srt.qz, srt.qw};
87
88     // normalize
89     const float  inv_sql = 1.f / (srt.qx * srt.qx + srt.qy * srt.qy + srt.qz * srt.qz + srt.qw * srt.qw);
90     const float4 nq      = optixMulFloat4(q, inv_sql);
91
92     const float sqw = q.w * nq.w;
93     const float sqx = q.x * nq.x;
94     const float sqy = q.y * nq.y;
95     const float sqz = q.z * nq.z;
96
97     const float xy = q.x * nq.y;
98     const float zw = q.z * nq.w;
99     const float xz = q.x * nq.z;
100     const float yw = q.y * nq.w;
101     const float yz = q.y * nq.z;
102     const float xw = q.x * nq.w;
103
104     m0.x = (sqx - sqy - sqz + sqw);
105     m0.y = 2.0f * (xy - zw);
106     m0.z = 2.0f * (xz + yw);
107
108     m1.x = 2.0f * (xy + zw);
109     m1.y = (-sqx + sqy - sqz + sqw);
110     m1.z = 2.0f * (yz - xw);
111
112     m2.x = 2.0f * (xz - yw);
113     m2.y = 2.0f * (yz + xw);
114     m2.z = (-sqx - sqy + sqz + sqw);
115
116     m0.w = m0.x * srt.pvx + m0.y * srt.pvy + m0.z * srt.pvz + srt.tx;
117     m1.w = m1.x * srt.pvx + m1.y * srt.pvy + m1.z * srt.pvz + srt.ty;
118     m2.w = m2.x * srt.pvx + m2.y * srt.pvy + m2.z * srt.pvz + srt.tz;
119
120     m0.z = m0.x * srt.b + m0.y * srt.c + m0.z * srt.sz;
121     m1.z = m1.x * srt.b + m1.y * srt.c + m1.z * srt.sz;
122     m2.z = m2.x * srt.b + m2.y * srt.c + m2.z * srt.sz;
123
124     m0.y = m0.x * srt.a + m0.y * srt.sy;
125     m1.y = m1.x * srt.a + m1.y * srt.sy;
126     m2.y = m2.x * srt.a + m2.y * srt.sy;
127
128     m0.x = m0.x * srt.sx;
129     m1.x = m1.x * srt.sx;
130     m2.x = m2.x * srt.sx;
131 }
132
133 // Inverts a 3x4 matrix in place
134 static __forceinline__ __device__ void optixInvertMatrix(float4& m0, float4& m1, float4& m2)
135 {
136     const float det3 =
137         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);
138
139     const float inv_det3 = 1.0f / det3;
140
141     float inv3[3][3];
142     inv3[0][0] = inv_det3 * (m1.y * m2.z - m2.y * m1.z);
143     inv3[0][1] = inv_det3 * (m0.z * m2.y - m2.z * m0.y);
144     inv3[0][2] = inv_det3 * (m0.y * m1.z - m1.y * m0.z);
145
146     inv3[1][0] = inv_det3 * (m1.z * m2.x - m2.z * m1.x);
147     inv3[1][1] = inv_det3 * (m0.x * m2.z - m2.x * m0.z);
```

```
148     inv3[1][2] = inv_det3 * (m0.z * m1.x - m1.z * m0.x);
149
150     inv3[2][0] = inv_det3 * (m1.x * m2.y - m2.x * m1.y);
151     inv3[2][1] = inv_det3 * (m0.y * m2.x - m2.y * m0.x);
152     inv3[2][2] = inv_det3 * (m0.x * m1.y - m1.x * m0.y);
153
154     const float b[3] = {m0.w, m1.w, m2.w};
155
156     m0.x = inv3[0][0];
157     m0.y = inv3[0][1];
158     m0.z = inv3[0][2];
159     m0.w = -inv3[0][0] * b[0] - inv3[0][1] * b[1] - inv3[0][2] * b[2];
160
161     m1.x = inv3[1][0];
162     m1.y = inv3[1][1];
163     m1.z = inv3[1][2];
164     m1.w = -inv3[1][0] * b[0] - inv3[1][1] * b[1] - inv3[1][2] * b[2];
165
166     m2.x = inv3[2][0];
167     m2.y = inv3[2][1];
168     m2.z = inv3[2][2];
169     m2.w = -inv3[2][0] * b[0] - inv3[2][1] * b[1] - inv3[2][2] * b[2];
170 }
171
172 static __forceinline__ __device__ void optixLoadInterpolatedMatrixKey(float4& m0, float4& m1, float4&
m2, const float4* matrix, const float t1)
173 {
174     m0 = optixLoadReadOnlyAlign16(&matrix[0]);
175     m1 = optixLoadReadOnlyAlign16(&matrix[1]);
176     m2 = optixLoadReadOnlyAlign16(&matrix[2]);
177
178     // The conditional prevents concurrent loads leading to spills
179     if(t1 > 0.0f)
180     {
181         const float t0 = 1.0f - t1;
182         m0 = optixAddFloat4(optixMulFloat4(m0, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[3]),
t1));
183         m1 = optixAddFloat4(optixMulFloat4(m1, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[4]),
t1));
184         m2 = optixAddFloat4(optixMulFloat4(m2, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[5]),
t1));
185     }
186 }
187
188 static __forceinline__ __device__ void optixLoadInterpolatedSrtKey(float4&       srt0,
189                                                                     float4&       srt1,
190                                                                     float4&       srt2,
191                                                                     float4&       srt3,
192                                                                     const float4* srt,
193                                                                     const float   t1)
194 {
195     srt0 = optixLoadReadOnlyAlign16(&srt[0]);
196     srt1 = optixLoadReadOnlyAlign16(&srt[1]);
197     srt2 = optixLoadReadOnlyAlign16(&srt[2]);
198     srt3 = optixLoadReadOnlyAlign16(&srt[3]);
199
200     // The conditional prevents concurrent loads leading to spills
201     if(t1 > 0.0f)
202     {
203         const float t0 = 1.0f - t1;
204         srt0 = optixAddFloat4(optixMulFloat4(srt0, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[4]),
t1));
205         srt1 = optixAddFloat4(optixMulFloat4(srt1, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[5]),
t1));
206         srt2 = optixAddFloat4(optixMulFloat4(srt2, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[6]),
t1));
207         srt3 = optixAddFloat4(optixMulFloat4(srt3, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[7]),
```

```
t1));
208
209         float inv_length = 1.f / sqrt(srt2.y * srt2.y + srt2.z * srt2.z + srt2.w * srt2.w + srt3.x *
srt3.x);
210         srt2.y *= inv_length;
211         srt2.z *= inv_length;
212         srt2.w *= inv_length;
213         srt3.x *= inv_length;
214     }
215 }
216
217 static __forceinline__ __device__ void optixResolveMotionKey(float& localt, int& key, const
OptixMotionOptions& options, const float globalt)
218 {
219     const float timeBegin    = options.timeBegin;
220     const float timeEnd      = options.timeEnd;
221     const float numIntervals = (float)(options.numKeys - 1);
222
223     // No need to check the motion flags. If data originates from a valid transform list handle, then
globalt is in
224     // range, or vanish flags are not set.
225
226     // should be NaN or in [0,numIntervals]
227     float time = max(0.f, min(numIntervals, numIntervals * __fdividef(globalt - timeBegin, timeEnd -
timeBegin)));
228
229     // catch NaN (for example when timeBegin=timeEnd)
230     if(time != time)
231         time = 0.f;
232
233     const float fltKey = fminf(floorf(time), numIntervals - 1);
234
235     localt = time - fltKey;
236     key    = (int)fltKey;
237 }
238
239 // Returns the interpolated transformation matrix for a particular matrix motion transformation and point
in time.
240 static __forceinline__ __device__ void optixGetInterpolatedTransformation(float4&
trf0,
241                                                                            float4&                      trf1,
242                                                                            float4&                      trf2,
243                                                                               const OptixMatrixMotionTransform*
transformData,
244                                                                            const float                  time)
245 {
246     // Compute key and intra key time
247     float keyTime;
248     int   key;
249     optixResolveMotionKey(keyTime, key, optixLoadReadOnlyAlign16(transformData).motionOptions, time);
250
251     // Get pointer to left key
252     const float4* transform = (const float4*)(&transformData->transform[key][0]);
253
254     // Load and interpolate matrix keys
255     optixLoadInterpolatedMatrixKey(trf0, trf1, trf2, transform, keyTime);
256 }
257
258 // Returns the interpolated transformation matrix for a particular SRT motion transformation and point in
time.
259 static __forceinline__ __device__ void optixGetInterpolatedTransformation(float4&
trf0,
260                                                                              float4&                      trf1,
261                                                                              float4&                      trf2,
262                                                                                  const OptixSRTMotionTransform*
transformData,
263                                                                              const float                  time)
```

```
264 {
265     // Compute key and intra key time
266     float keyTime;
267     int   key;
268     optixResolveMotionKey(keyTime, key, optixLoadReadOnlyAlign16(transformData).motionOptions, time);
269
270     // Get pointer to left key
271     const float4* dataPtr = reinterpret_cast<const float4*>(&transformData->srtData[key]);
272
273     // Load and interpolated SRT keys
274     float4 data[4];
275     optixLoadInterpolatedSrtKey(data[0], data[1], data[2], data[3], dataPtr, keyTime);
276
277     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,
278                         data[2].x, data[2].y, data[2].z, data[2].w, data[3].x, data[3].y, data[3].z,
data[3].w};
279
280     // Convert SRT into a matrix
281     optixGetMatrixFromSrt(trf0, trf1, trf2, srt);
282 }
283
284 // Returns the interpolated transformation matrix for a particular traversable handle and point in time.
285 static __forceinline__ __device__ void optixGetInterpolatedTransformationFromHandle(float4&
trf0,
286                                                                                     float4&
trf1,
287                                                                                     float4&
trf2,
288                                                                                     const
OptixTraversableHandle handle,
289                                                                                     const float
time,
290                                                                                   const bool objectToWorld)
291 {
292     const OptixTransformType type = optixGetTransformTypeFromHandle(handle);
293
294     if(type == OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM || type ==
OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM)
295     {
296         if(type == OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM)
297         {
298             const OptixMatrixMotionTransform* transformData =
optixGetMatrixMotionTransformFromHandle(handle);
299             optixGetInterpolatedTransformation(trf0, trf1, trf2, transformData, time);
300         }
301         else
302         {
303             const OptixSRTMotionTransform* transformData = optixGetSRTMotionTransformFromHandle(handle);
304             optixGetInterpolatedTransformation(trf0, trf1, trf2, transformData, time);
305         }
306
307         if(!objectToWorld)
308             optixInvertMatrix(trf0, trf1, trf2);
309     }
310     else if(type == OPTIX_TRANSFORM_TYPE_INSTANCE || type == OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM)
311     {
312         const float4* transform;
313
314         if(type == OPTIX_TRANSFORM_TYPE_INSTANCE)
315         {
316             transform = (objectToWorld) ? optixGetInstanceTransformFromHandle(handle) :
317                                           optixGetInstanceInverseTransformFromHandle(handle);
318         }
319         else
320         {
321             const OptixStaticTransform* traversable = optixGetStaticTransformFromHandle(handle);
```

```
322             transform = (const float4*)((objectToWorld) ? traversable->transform :
traversable->invTransform);
323         }
324
325         trf0 = optixLoadReadOnlyAlign16(&transform[0]);
326         trf1 = optixLoadReadOnlyAlign16(&transform[1]);
327         trf2 = optixLoadReadOnlyAlign16(&transform[2]);
328     }
329     else
330     {
331         trf0 = {1.0f, 0.0f, 0.0f, 0.0f};
332         trf1 = {0.0f, 1.0f, 0.0f, 0.0f};
333         trf2 = {0.0f, 0.0f, 1.0f, 0.0f};
334     }
335 }
336
337 // Returns the world-to-object transformation matrix resulting from the current transform stack and
current ray time.
338 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float4& m0, float4& m1,
float4& m2)
339 {
340     const unsigned int size = optixGetTransformListSize();
341     const float        time = optixGetRayTime();
342
343 #pragma unroll 1
344     for(unsigned int i = 0; i < size; ++i)
345     {
346         OptixTraversableHandle handle = optixGetTransformListHandle(i);
347
348         float4 trf0, trf1, trf2;
349         optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
false);
350
351         if(i == 0)
352         {
353             m0 = trf0;
354             m1 = trf1;
355             m2 = trf2;
356         }
357         else
358         {
359             // m := trf * m
360             float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
361             m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
362             m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
363             m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
364         }
365     }
366 }
367
368 // Returns the object-to-world transformation matrix resulting from the current transform stack and
current ray time.
369 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float4& m0, float4& m1,
float4& m2)
370 {
371     const int   size = optixGetTransformListSize();
372     const float time = optixGetRayTime();
373
374 #pragma unroll 1
375     for(int i = size - 1; i >= 0; --i)
376     {
377         OptixTraversableHandle handle = optixGetTransformListHandle(i);
378
379         float4 trf0, trf1, trf2;
380         optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
true);
381
```

```
382         if(i == size - 1)
383         {
384             m0 = trf0;
385             m1 = trf1;
386             m2 = trf2;
387         }
388         else
389         {
390             // m := trf * m
391             float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
392             m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
393             m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
394             m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
395         }
396     }
397 }
398
399 // Multiplies the 3x4 matrix with rows m0, m1, m2 with the point p.
400 static __forceinline__ __device__ float3 optixTransformPoint(const float4& m0, const float4& m1, const
float4& m2, const float3& p)
401 {
402     float3 result;
403     result.x = m0.x * p.x + m0.y * p.y + m0.z * p.z + m0.w;
404     result.y = m1.x * p.x + m1.y * p.y + m1.z * p.z + m1.w;
405     result.z = m2.x * p.x + m2.y * p.y + m2.z * p.z + m2.w;
406     return result;
407 }
408
409 // Multiplies the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the vector v.
410 static __forceinline__ __device__ float3 optixTransformVector(const float4& m0, const float4& m1, const
float4& m2, const float3& v)
411 {
412     float3 result;
413     result.x = m0.x * v.x + m0.y * v.y + m0.z * v.z;
414     result.y = m1.x * v.x + m1.y * v.y + m1.z * v.z;
415     result.z = m2.x * v.x + m2.y * v.y + m2.z * v.z;
416     return result;
417 }
418
419 // Multiplies the transpose of the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the
normal n.
420 // Note that the given matrix is supposed to be the inverse of the actual transformation matrix.
421 static __forceinline__ __device__ float3 optixTransformNormal(const float4& m0, const float4& m1, const
float4& m2, const float3& n)
422 {
423     float3 result;
424     result.x = m0.x * n.x + m1.x * n.y + m2.x * n.z;
425     result.y = m0.y * n.x + m1.y * n.y + m2.y * n.z;
426     result.z = m0.z * n.x + m1.z * n.y + m2.z * n.z;
427     return result;
428 }
429
430 }  // namespace optix_impl
431
432 #endif // OPTIX_OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H
```

8.5 optix_micromap_impl.h File Reference

Namespaces

- namespace optix_impl

Macros

- #define OPTIX_MICROMAP_FUNC
- #define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline

- #define OPTIX_MICROMAP_FLOAT2_SUB(a, b) { a.x - b.x, a.y - b.y }

Functions

- OPTIX_MICROMAP_INLINE_FUNC float optix_impl::__uint_as_float (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::extractEvenBits (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::prefixEor (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC void optix_impl::index2dbary (unsigned int index, unsigned int &u, unsigned int &v, unsigned int &w)
- OPTIX_MICROMAP_INLINE_FUNC void optix_impl::micro2bary (unsigned int index, unsigned int subdivisionLevel, float2 &bary0, float2 &bary1, float2 &bary2)
- OPTIX_MICROMAP_INLINE_FUNC float2 optix_impl::base2micro (const float2 &baseBarycentrics, const float2 microVertexBaseBarycentrics[3])

8.5.1 Detailed Description

OptiX micromap helper functions.

Author

NVIDIA Corporation

8.5.2 Macro Definition Documentation

8.5.2.1 OPTIX_MICROMAP_FUNC

```
#define OPTIX_MICROMAP_FUNC
```

8.6 optix_micromap_impl.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2023 NVIDIA Corporation.  All rights reserved.
3 *
4 * Redistribution and use in source and binary forms, with or without
5 * modification, are permitted provided that the following conditions
6 * are met:
7 *  * Redistributions of source code must retain the above copyright
8 *    notice, this list of conditions and the following disclaimer.
9 *  * Redistributions in binary form must reproduce the above copyright
10 *    notice, this list of conditions and the following disclaimer in the
11 *    documentation and/or other materials provided with the distribution.
12 *  * Neither the name of NVIDIA CORPORATION nor the names of its
13 *    contributors may be used to endorse or promote products derived
14 *    from this software without specific prior written permission.
15 *
16 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS “AS IS” AND ANY
17 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
18 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
19 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
20 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
21 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
22 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
23 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
24 * OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
25 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
26 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
27 */
28
29
```

```
36 #ifndef OPTIX_OPTIX_MICROMAP_IMPL_H
37 #define OPTIX_OPTIX_MICROMAP_IMPL_H
38
39 #ifndef OPTIX_MICROMAP_FUNC
40 #ifdef __CUDACC__
41 #define OPTIX_MICROMAP_FUNC __device__
42 #else
43 #define OPTIX_MICROMAP_FUNC
44 #endif
45 #endif
46
47 namespace optix_impl {
48
53 #define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline
54
55 #ifdef __CUDACC__
56 // the device implementation of __uint_as_float is declared in cuda_runtime.h
57 #else
58 // the host implementation of __uint_as_float
59 OPTIX_MICROMAP_INLINE_FUNC float __uint_as_float(unsigned int x)
60 {
61     union { float f; unsigned int i; } var;
62     var.i = x;
63     return var.f;
64 }
65 #endif
66
67 // Extract even bits
68 OPTIX_MICROMAP_INLINE_FUNC unsigned int extractEvenBits(unsigned int x)
69 {
70     x &= 0x55555555;
71     x = (x | (x » 1)) & 0x33333333;
72     x = (x | (x » 2)) & 0x0f0f0f0f;
73     x = (x | (x » 4)) & 0x00ff00ff;
74     x = (x | (x » 8)) & 0x0000ffff;
75     return x;
76 }
77
78
79 // Calculate exclusive prefix or (log(n) XOR's and SHF's)
80 OPTIX_MICROMAP_INLINE_FUNC unsigned int prefixEor(unsigned int x)
81 {
82     x ^= x » 1;
83     x ^= x » 2;
84     x ^= x » 4;
85     x ^= x » 8;
86     return x;
87 }
88
89 // Convert distance along the curve to discrete barycentrics
90 OPTIX_MICROMAP_INLINE_FUNC void index2dbary(unsigned int index, unsigned int& u, unsigned int& v, unsigned
int& w)
91 {
92     unsigned int b0 = extractEvenBits(index);
93     unsigned int b1 = extractEvenBits(index » 1);
94
95     unsigned int fx = prefixEor(b0);
96     unsigned int fy = prefixEor(b0 & ~b1);
97
98     unsigned int t = fy ^ b1;
99
100     u = (fx & ~t) | (b0 & ~t) | (~b0 & ~fx & t);
101     v = fy ^ b0;
102     w = (~fx & ~t) | (b0 & ~t) | (~b0 & fx & t);
103 }
104
105 // Compute barycentrics of a sub or micro triangle wrt a base triangle.  The order of the returned
```

```
106 // bary0, bary1, bary2 matters and allows for using this function for sub triangles and the
107 // conversion from sub triangle to base triangle barycentric space
108 OPTIX_MICROMAP_INLINE_FUNC void micro2bary(unsigned int index, unsigned int subdivisionLevel, float2&
bary0, float2& bary1, float2& bary2)
109 {
110     if(subdivisionLevel == 0)
111     {
112         bary0 = { 0, 0 };
113         bary1 = { 1, 0 };
114         bary2 = { 0, 1 };
115         return;
116     }
117
118     unsigned int iu, iv, iw;
119     index2dbary(index, iu, iv, iw);
120
121     // we need to only look at "level" bits
122     iu = iu & ((1 « subdivisionLevel) - 1);
123     iv = iv & ((1 « subdivisionLevel) - 1);
124     iw = iw & ((1 « subdivisionLevel) - 1);
125
126     int yFlipped = (iu & 1) ^ (iv & 1) ^ (iw & 1) ^ 1;
127
128     int xFlipped = ((0x8888888888888888ull ^ 0xf000f000f000f000ull ^ 0xffff000000000000ull) » index) & 1;
129     xFlipped    ^= ((0x8888888888888888ull ^ 0xf000f000f000f000ull ^ 0xffff000000000000ull) » (index »
6)) & 1;
130
131     const float levelScale = __uint_as_float((127u - subdivisionLevel) « 23);
132
133     // scale the barycentic coordinate to the global space/scale
134     float du = 1.f * levelScale;
135     float dv = 1.f * levelScale;
136
137     // scale the barycentic coordinate to the global space/scale
138     float u = (float)iu * levelScale;
139     float v = (float)iv * levelScale;
140
141     //     c        d
142     //      x-----x
143     //     / \   /
144     //    /   \ /
145     //   x-----x
146     //  a        b
147     //
148     // !xFlipped && !yFlipped: abc
149     // !xFlipped &&  yFlipped: cdb
150     //  xFlipped && !yFlipped: bac
151     //  xFlipped &&  yFlipped: dcb
152
153     bary0 = { u + xFlipped * du    , v + yFlipped * dv };
154     bary1 = { u + (1-xFlipped) * du, v + yFlipped * dv };
155     bary2 = { u + yFlipped * du    , v + (1-yFlipped) * dv };
156 }
157
158 // avoid any conflicts due to multiple definitions
159 #define OPTIX_MICROMAP_FLOAT2_SUB(a,b) { a.x - b.x, a.y - b.y }
160
161 // Compute barycentrics for micro triangle from base barycentrics
162 OPTIX_MICROMAP_INLINE_FUNC float2 base2micro(const float2& baseBarycentrics, const float2
microVertexBaseBarycentrics[3])
163 {
164     float2 baryV0P  = OPTIX_MICROMAP_FLOAT2_SUB(baseBarycentrics, microVertexBaseBarycentrics[0]);
165     float2 baryV0V1 = OPTIX_MICROMAP_FLOAT2_SUB(microVertexBaseBarycentrics[1],
microVertexBaseBarycentrics[0]);
166     float2 baryV0V2 = OPTIX_MICROMAP_FLOAT2_SUB(microVertexBaseBarycentrics[2],
microVertexBaseBarycentrics[0]);
167
```

```
168     float  rdetA = 1.f / (baryV0V1.x * baryV0V2.y - baryV0V1.y * baryV0V2.x);
169     float4 A     = { baryV0V2.y, -baryV0V2.x, -baryV0V1.y, baryV0V1.x };
170
171     float2 localUV;
172     localUV.x = rdetA * (baryV0P.x * A.x + baryV0P.y * A.y);
173     localUV.y = rdetA * (baryV0P.x * A.z + baryV0P.y * A.w);
174
175     return localUV;
176 }
177 #undef OPTIX_MICROMAP_FLOAT2_SUB
178   // end group optix_utilities
180
181 }  // namespace optix_impl
182
183 #endif  // OPTIX_OPTIX_MICROMAP_IMPL_H
```

8.7 optix.h File Reference

Macros

- #define OPTIX_VERSION 80000

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 80000
```

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) 2023 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 80000
38
39
40 #ifdef __CUDACC__
41 #include "optix_device.h"
42 #else
43 #include "optix_host.h"
44 #endif
45
46
47 #endif  // OPTIX_OPTIX_H
```

8.9 optix_denoiser_tiling.h File Reference

Classes

- struct OptixUtilDenoiserImageTile

Functions

- OptixResult optixUtilGetPixelStride (const OptixImage2D &image, unsigned int &pixelStrideInBytes)
- OptixResult optixUtilDenoiserSplitImage (const OptixImage2D &input, const OptixImage2D &output, unsigned int overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight, std::vector< OptixUtilDenoiserImageTile > &tiles)
- OptixResult optixUtilDenoiserInvokeTiled (OptixDenoiser denoiser, CUstream stream, const OptixDenoiserParams *params, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, const OptixDenoiserGuideLayer *guideLayer, const OptixDenoiserLayer *layers, unsigned int numLayers, CUdeviceptr scratch, size_t scratchSizeInBytes, unsigned int overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight)

8.9.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.10 optix_denoiser_tiling.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2023 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
73 inline OptixResult optixUtilGetPixelStride(const OptixImage2D& image, unsigned int& pixelStrideInBytes)
74 {
75     pixelStrideInBytes = image.pixelStrideInBytes;
76     if(pixelStrideInBytes == 0)
77     {
78         switch(image.format)
79         {
80             case OPTIX_PIXEL_FORMAT_HALF1:
81                 pixelStrideInBytes = 1 * sizeof(short);
82                 break;
83             case OPTIX_PIXEL_FORMAT_HALF2:
84                 pixelStrideInBytes = 2 * sizeof(short);
85                 break;
86             case OPTIX_PIXEL_FORMAT_HALF3:
87                 pixelStrideInBytes = 3 * sizeof(short);
88                 break;
89             case OPTIX_PIXEL_FORMAT_HALF4:
90                 pixelStrideInBytes = 4 * sizeof(short);
91                 break;
92             case OPTIX_PIXEL_FORMAT_FLOAT1:
```

```
93                 pixelStrideInBytes = 1 * sizeof(float);
94                 break;
95             case OPTIX_PIXEL_FORMAT_FLOAT2:
96                 pixelStrideInBytes = 2 * sizeof(float);
97                 break;
98             case OPTIX_PIXEL_FORMAT_FLOAT3:
99                 pixelStrideInBytes = 3 * sizeof(float);
100                break;
101            case OPTIX_PIXEL_FORMAT_FLOAT4:
102                pixelStrideInBytes = 4 * sizeof(float);
103                break;
104            case OPTIX_PIXEL_FORMAT_UCHAR3:
105                pixelStrideInBytes = 3 * sizeof(char);
106                break;
107            case OPTIX_PIXEL_FORMAT_UCHAR4:
108                pixelStrideInBytes = 4 * sizeof(char);
109                break;
110            case OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER:
111                return OPTIX_ERROR_INVALID_VALUE;
112                break;
113        }
114    }
115    return OPTIX_SUCCESS;
116 }
117
127 inline OptixResult optixUtilDenoiserSplitImage(
128                                                 const OptixImage2D&                    input,
129                                                 const OptixImage2D&                    output,
130                                           unsigned int                         overlapWindowSizeInPixels,
131                                                 unsigned int                           tileWidth,
132                                                 unsigned int                           tileHeight,
133                                                 std::vector<OptixUtilDenoiserImageTile>&    tiles)
134 {
135    if(tileWidth == 0 || tileHeight == 0)
136        return OPTIX_ERROR_INVALID_VALUE;
137
138    unsigned int inPixelStride, outPixelStride;
139    if(const OptixResult res = optixUtilGetPixelStride(input, inPixelStride))
140        return res;
141    if(const OptixResult res = optixUtilGetPixelStride(output, outPixelStride))
142        return res;
143
144    int inp_w = std::min(tileWidth + 2 * overlapWindowSizeInPixels, input.width);
145    int inp_h = std::min(tileHeight + 2 * overlapWindowSizeInPixels, input.height);
146    int inp_y = 0, copied_y = 0;
147
148    int upscaleX = output.width / input.width;
149    int upscaleY = output.height / input.height;
150
151    do
152    {
153        int inputOffsetY = inp_y == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_h -
((int)input.height - inp_y));
154        int copy_y       = inp_y == 0 ? std::min(input.height, tileHeight + overlapWindowSizeInPixels) :
155                                        std::min(tileHeight, input.height - copied_y);
156
157        int inp_x = 0, copied_x = 0;
158        do
159        {
160            int inputOffsetX = inp_x == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_w -
((int)input.width - inp_x));
161            int copy_x = inp_x == 0 ? std::min(input.width, tileWidth + overlapWindowSizeInPixels) :
162                                      std::min(tileWidth, input.width - copied_x);
163
164            OptixUtilDenoiserImageTile tile;
165            tile.input.data              = input.data + (size_t)(inp_y - inputOffsetY) *
input.rowStrideInBytes
```

```
166                                         + (size_t)(inp_x - inputOffsetX) * inPixelStride;
167             tile.input.width              = inp_w;
168             tile.input.height             = inp_h;
169             tile.input.rowStrideInBytes   = input.rowStrideInBytes;
170             tile.input.pixelStrideInBytes = input.pixelStrideInBytes;
171             tile.input.format             = input.format;
172
173             tile.output.data               = output.data + (size_t)(upscaleY * inp_y) *
output.rowStrideInBytes
174                                            + (size_t)(upscaleX * inp_x) * outPixelStride;
175             tile.output.width              = upscaleX * copy_x;
176             tile.output.height             = upscaleY * copy_y;
177             tile.output.rowStrideInBytes   = output.rowStrideInBytes;
178             tile.output.pixelStrideInBytes = output.pixelStrideInBytes;
179             tile.output.format             = output.format;
180
181             tile.inputOffsetX = inputOffsetX;
182             tile.inputOffsetY = inputOffsetY;
183
184             tiles.push_back(tile);
185
186             inp_x += inp_x == 0 ? tileWidth + overlapWindowSizeInPixels : tileWidth;
187             copied_x += copy_x;
188         } while(inp_x < static_cast<int>(input.width));
189
190         inp_y += inp_y == 0 ? tileHeight + overlapWindowSizeInPixels : tileHeight;
191         copied_y += copy_y;
192     } while(inp_y < static_cast<int>(input.height));
193
194     return OPTIX_SUCCESS;
195 }
196
200
207
223 inline OptixResult optixUtilDenoiserInvokeTiled(
224                                                 OptixDenoiser                   denoiser,
225                                                 CUstream                        stream,
226                                                 const OptixDenoiserParams*      params,
227                                                 CUdeviceptr                     denoiserState,
228                                                 size_t                         denoiserStateSizeInBytes,
229                                                 const OptixDenoiserGuideLayer*  guideLayer,
230                                                 const OptixDenoiserLayer*       layers,
231                                                 unsigned int                    numLayers,
232                                                 CUdeviceptr                     scratch,
233                                                 size_t                          scratchSizeInBytes,
234                                                unsigned int                    overlapWindowSizeInPixels,
235                                                 unsigned int                    tileWidth,
236                                                 unsigned int                    tileHeight)
237 {
238     if(!guideLayer || !layers)
239         return OPTIX_ERROR_INVALID_VALUE;
240
241     const unsigned int upscale = numLayers > 0 && layers[0].previousOutput.width == 2 *
layers[0].input.width ? 2 : 1;
242
243     std::vector<std::vector<OptixUtilDenoiserImageTile» tiles(numLayers);
244     std::vector<std::vector<OptixUtilDenoiserImageTile» prevTiles(numLayers);
245     for(unsigned int l = 0; l < numLayers; l++)
246     {
247         if(const OptixResult res = optixUtilDenoiserSplitImage(layers[l].input, layers[l].output,
248                                                                overlapWindowSizeInPixels,
249                                                                tileWidth, tileHeight, tiles[l]))
250             return res;
251
252         if(layers[l].previousOutput.data)
253         {
254             OptixImage2D dummyOutput = layers[l].previousOutput;
```

```
255             if(const OptixResult res = optixUtilDenoiserSplitImage(layers[l].previousOutput, dummyOutput,
256                                                                    upscale * overlapWindowSizeInPixels,
257                                                                   upscale * tileWidth, upscale * tileHeight,
prevTiles[l]))
258                 return res;
259         }
260     }
261
262     std::vector<OptixUtilDenoiserImageTile> albedoTiles;
263     if(guideLayer->albedo.data)
264     {
265         OptixImage2D dummyOutput = guideLayer->albedo;
266         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->albedo, dummyOutput,
267                                                                overlapWindowSizeInPixels,
268                                                                tileWidth, tileHeight, albedoTiles))
269             return res;
270     }
271
272     std::vector<OptixUtilDenoiserImageTile> normalTiles;
273     if(guideLayer->normal.data)
274     {
275         OptixImage2D dummyOutput = guideLayer->normal;
276         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->normal, dummyOutput,
277                                                                overlapWindowSizeInPixels,
278                                                                tileWidth, tileHeight, normalTiles))
279             return res;
280     }
281
282     std::vector<OptixUtilDenoiserImageTile> flowTiles;
283     if(guideLayer->flow.data)
284     {
285         OptixImage2D dummyOutput = guideLayer->flow;
286         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->flow, dummyOutput,
287                                                                overlapWindowSizeInPixels,
288                                                                tileWidth, tileHeight, flowTiles))
289             return res;
290     }
291
292     std::vector<OptixUtilDenoiserImageTile> flowTrustTiles;
293     if(guideLayer->flowTrustworthiness.data)
294     {
295         OptixImage2D dummyOutput = guideLayer->flowTrustworthiness;
296         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->flowTrustworthiness,
dummyOutput,
297                                                                overlapWindowSizeInPixels,
298                                                                tileWidth, tileHeight, flowTrustTiles))
299             return res;
300     }
301
302     std::vector<OptixUtilDenoiserImageTile> internalGuideLayerTiles;
303     if(guideLayer->previousOutputInternalGuideLayer.data && guideLayer->outputInternalGuideLayer.data)
304     {
305         if(const OptixResult res =
optixUtilDenoiserSplitImage(guideLayer->previousOutputInternalGuideLayer,
306                                                                guideLayer->outputInternalGuideLayer,
307                                                                upscale * overlapWindowSizeInPixels,
308                                                               upscale * tileWidth, upscale * tileHeight,
internalGuideLayerTiles))
309             return res;
310     }
311
312     for(size_t t = 0; t < tiles[0].size(); t++)
313     {
314         std::vector<OptixDenoiserLayer> tlayers;
315         for(unsigned int l = 0; l < numLayers; l++)
316         {
317             OptixDenoiserLayer layer = {};
```

```
318             layer.input  = (tiles[l])[t].input;
319             layer.output = (tiles[l])[t].output;
320             if(layers[l].previousOutput.data)
321                 layer.previousOutput = (prevTiles[l])[t].input;
322             layer.type = layers[l].type;
323             tlayers.push_back(layer);
324         }
325 
326         OptixDenoiserGuideLayer gl = {};
327         if(guideLayer->albedo.data)
328             gl.albedo = albedoTiles[t].input;
329 
330         if(guideLayer->normal.data)
331             gl.normal = normalTiles[t].input;
332 
333         if(guideLayer->flow.data)
334             gl.flow = flowTiles[t].input;
335 
336         if(guideLayer->flowTrustworthiness.data)
337             gl.flowTrustworthiness = flowTrustTiles[t].input;
338 
339         if(guideLayer->previousOutputInternalGuideLayer.data)
340             gl.previousOutputInternalGuideLayer = internalGuideLayerTiles[t].input;
341 
342         if(guideLayer->outputInternalGuideLayer.data)
343             gl.outputInternalGuideLayer = internalGuideLayerTiles[t].output;
344 
345         if(const OptixResult res =
346                 optixDenoiserInvoke(denoiser, stream, params, denoiserState, denoiserStateSizeInBytes,
347                                     &gl, &tlayers[0], numLayers,
348                                     (tiles[0])[t].inputOffsetX, (tiles[0])[t].inputOffsetY,
349                                     scratch, scratchSizeInBytes))
350             return res;
351     }
352     return OPTIX_SUCCESS;
353 }
354   // end group optix_utilities
356 
357 #ifdef __cplusplus
358 }
359 #endif
360 
361 #endif  // OPTIX_DENOISER_TILING_H
```

8.11 optix_device.h File Reference

Macros

- #define __OPTIX_INCLUDE_INTERNAL_HEADERS__

Functions

- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTraverse (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>

static __forceinline__ __device__ void optixTrace (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)

- template<typename... Payload>
  static __forceinline__ __device__ void optixTraverse (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- static __forceinline__ __device__ void optixReorder (unsigned int coherenceHint, unsigned int numCoherenceHintBitsFromLSB)
- static __forceinline__ __device__ void optixReorder ()
- template<typename... Payload>
  static __forceinline__ __device__ void optixInvoke (Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixInvoke (OptixPayloadTypeID type, Payload &... payload)
- template<typename... RegAttributes>
  static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int SBToffset, unsigned int SBTstride, unsigned int instIdx, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)
- template<typename... RegAttributes>
  static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int SBToffset, unsigned int SBTstride, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)
- template<typename... RegAttributes>
  static __forceinline__ __device__ void optixMakeHitObjectWithRecord (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int sbtRecordIndex, unsigned int instIdx, const OptixTraversableHandle *transforms, unsigned int numTransforms, unsigned int sbtGASIdx, unsigned int primIdx, unsigned int hitKind, RegAttributes... regAttributes)
- static __forceinline__ __device__ void optixMakeMissHitObject (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime)
- static __forceinline__ __device__ void optixMakeNopHitObject ()
- static __forceinline__ __device__ bool optixHitObjectIsHit ()
- static __forceinline__ __device__ bool optixHitObjectIsMiss ()
- static __forceinline__ __device__ bool optixHitObjectIsNop ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex ()
- static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_5 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_6 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_7 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_8 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_9 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_10 (unsigned int p)

- static __forceinline__ __device__ void optixSetPayload_11 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_12 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_13 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_14 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_15 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_16 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_17 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_18 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_19 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_20 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_21 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_22 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_23 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_24 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_25 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_26 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_27 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_28 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_29 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_30 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_31 (unsigned int p)
- static __forceinline__ __device__ unsigned int optixGetPayload_0 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_1 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_2 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_3 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_4 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_5 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_6 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_7 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_8 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_9 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_10 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_11 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_12 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_13 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_14 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_15 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_16 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_17 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_18 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_19 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_20 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_21 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_22 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_23 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_24 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_25 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_26 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_27 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_28 ()

- static __forceinline__ __device__ unsigned int optixGetPayload_29 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_30 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_31 ()
- static __forceinline__ __device__ void optixSetPayloadTypes (unsigned int typeMask)
- static __forceinline__ __device__ unsigned int optixUndefinedValue ()
- static __forceinline__ __device__ float3 optixGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixGetObjectRayOrigin ()
- static __forceinline__ __device__ float3 optixGetObjectRayDirection ()
- static __forceinline__ __device__ float optixGetRayTmin ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmin ()
- static __forceinline__ __device__ float optixGetRayTmax ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmax ()
- static __forceinline__ __device__ float optixGetRayTime ()
- static __forceinline__ __device__ float optixHitObjectGetRayTime ()
- static __forceinline__ __device__ unsigned int optixGetRayFlags ()
- static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceTraversableFromIAS (OptixTraversableHandle ias, unsigned int instIdx)
- static __forceinline__ __device__ void optixGetTriangleVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static __forceinline__ __device__ void optixGetMicroTriangleVertexData (float3 data[3])
- static __forceinline__ __device__ void optixGetMicroTriangleBarycentricsData (float2 data[3])
- static __forceinline__ __device__ void optixGetLinearCurveVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetCubicBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetRibbonVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ float3 optixGetRibbonNormal (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ void optixGetSphereData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle ()
- static __forceinline__ __device__ float optixGetGASMotionTimeBegin (OptixTraversableHandle gas)
- static __forceinline__ __device__ float optixGetGASMotionTimeEnd (OptixTraversableHandle gas)
- static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (OptixTraversableHandle gas)
- static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (float m[12])

- static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (float3 normal)
- static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (float3 normal)
- static __forceinline__ __device__ unsigned int optixGetTransformListSize ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTransformType optixGetTransformTypeFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixStaticTransform * optixGetStaticTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixSRTMotionTransform * optixGetSRTMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixMatrixMotionTransform * optixGetMatrixMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceChildFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceInverseTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6)

- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6, unsigned int a7)
- static __forceinline__ __device__ unsigned int optixGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_7 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7 ()
- static __forceinline__ __device__ void optixTerminateRay ()
- static __forceinline__ __device__ void optixIgnoreIntersection ()
- static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixGetHitKind ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind ()
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsFrontFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsBackFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ()
- static __forceinline__ __device__ bool optixIsFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsBackFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleHit ()
- static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ()
- static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit ()
- static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleBackFaceHit ()
- static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ()
- static __forceinline__ __device__ float optixGetCurveParameter ()
- static __forceinline__ __device__ float2 optixGetRibbonParameters ()
- static __forceinline__ __device__ uint3 optixGetLaunchIndex ()
- static __forceinline__ __device__ uint3 optixGetLaunchDimensions ()
- static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ()

- static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer ()
- static __forceinline__ __device__ void optixThrowException (int exceptionCode)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6, unsigned int exceptionDetail7)
- static __forceinline__ __device__ int optixGetExceptionCode ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7 ()
- static __forceinline__ __device__ char * optixGetExceptionLineInfo ()
- template<typename ReturnT , typename... ArgTypes>
  static __forceinline__ __device__ ReturnT optixDirectCall (unsigned int sbtIndex, ArgTypes... args)
- template<typename ReturnT , typename... ArgTypes>
  static __forceinline__ __device__ ReturnT optixContinuationCall (unsigned int sbtIndex, ArgTypes... args)
- static __forceinline__ __device__ uint4 optixTexFootprint2D (unsigned long long tex, unsigned int texInfo, float x, float y, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DLod (unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool coarse, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DGrad (unsigned long long tex, unsigned int texInfo, float x, float y, float dPdx_x, float dPdx_y, float dPdy_x, float dPdy_y, bool coarse, unsigned int *singleMipLevel)

8.11.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

OptiX public API Reference - Device API declarations

8.11.2 Macro Definition Documentation

8.11.2.1 __OPTIX_INCLUDE_INTERNAL_HEADERS__

```
#define __OPTIX_INCLUDE_INTERNAL_HEADERS__
```

8.12 optix_device.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2023 NVIDIA Corporation.  All rights reserved.
3 *
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly
8 * prohibited.
9 *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
26
27 #ifndef OPTIX_OPTIX_DEVICE_H
28 #define OPTIX_OPTIX_DEVICE_H
29
30 #if defined(__cplusplus) && (__cplusplus < 201103L) && !defined(_WIN32)
31 #error Device code for OptiX requires at least C++11. Consider adding "--std c++11" to the nvcc
command-line.
32 #endif
33
34 #include "optix_types.h"
35
38
60 template <typename... Payload>
61 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
62                                                   float3                rayOrigin,
63                                                   float3                rayDirection,
64                                                   float                 tmin,
65                                                   float                 tmax,
66                                                   float                 rayTime,
67                                                   OptixVisibilityMask   visibilityMask,
68                                                   unsigned int          rayFlags,
69                                                   unsigned int          SBToffset,
70                                                   unsigned int          SBTstride,
71                                                   unsigned int          missSBTIndex,
72                                                   Payload&...             payload);
```

```
73
93 template <typename... Payload>
94 static __forceinline__ __device__ void optixTraverse(OptixTraversableHandle handle,
95                                                       float3                 rayOrigin,
96                                                       float3                 rayDirection,
97                                                       float                  tmin,
98                                                       float                  tmax,
99                                                       float                  rayTime,
100                                                       OptixVisibilityMask    visibilityMask,
101                                                       unsigned int           rayFlags,
102                                                       unsigned int           SBToffset,
103                                                       unsigned int           SBTstride,
104                                                       unsigned int           missSBTIndex,
105                                                       Payload&... payload);
106
124 template <typename... Payload>
125 static __forceinline__ __device__ void optixTrace(OptixPayloadTypeID     type,
126                                                    OptixTraversableHandle handle,
127                                                    float3                 rayOrigin,
128                                                    float3                 rayDirection,
129                                                    float                  tmin,
130                                                    float                  tmax,
131                                                    float                  rayTime,
132                                                    OptixVisibilityMask    visibilityMask,
133                                                    unsigned int           rayFlags,
134                                                    unsigned int           SBToffset,
135                                                    unsigned int           SBTstride,
136                                                    unsigned int           missSBTIndex,
137                                                    Payload&...              payload);
138
159 template <typename... Payload>
160 static __forceinline__ __device__ void optixTraverse(OptixPayloadTypeID     type,
161                                                       OptixTraversableHandle handle,
162                                                       float3                 rayOrigin,
163                                                       float3                 rayDirection,
164                                                       float                  tmin,
165                                                       float                  tmax,
166                                                       float                  rayTime,
167                                                       OptixVisibilityMask    visibilityMask,
168                                                       unsigned int           rayFlags,
169                                                       unsigned int           SBToffset,
170                                                       unsigned int           SBTstride,
171                                                       unsigned int           missSBTIndex,
172                                                       Payload&... payload);
173
184 static __forceinline__ __device__ void optixReorder(unsigned int coherenceHint, unsigned int
numCoherenceHintBitsFromLSB);
185
189 static __forceinline__ __device__ void optixReorder();
190
199 template <typename... Payload>
200 static __forceinline__ __device__ void optixInvoke(Payload&... payload);
201
211 template <typename... Payload>
212 static __forceinline__ __device__ void optixInvoke(OptixPayloadTypeID type, Payload&... payload);
213
232 template <typename... RegAttributes>
233 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle handle,
234                                                           float3                 rayOrigin,
235                                                           float3                 rayDirection,
236                                                           float                  tmin,
237                                                           float                  tmax,
238                                                           float                  rayTime,
239                                                           unsigned int           SBToffset,
240                                                           unsigned int           SBTstride,
241                                                           unsigned int           instIdx,
242                                                           unsigned int           sbtGASIdx,
```

```
243                                                     unsigned int           primIdx,
244                                                     unsigned int           hitKind,
245                                                     RegAttributes... regAttributes);
246
269 template <typename... RegAttributes>
270 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle        handle,
271                                                     float3                          rayOrigin,
272                                                     float3                          rayDirection,
273                                                     float                           tmin,
274                                                     float                           tmax,
275                                                     float                           rayTime,
276                                                     unsigned int                    SBToffset,
277                                                     unsigned int                    SBTstride,
278                                                     unsigned int                    instIdx,
279                                                     const OptixTraversableHandle* transforms,
280                                                     unsigned int                    numTransforms,
281                                                     unsigned int                    sbtGASIdx,
282                                                     unsigned int                    primIdx,
283                                                     unsigned int                    hitKind,
284                                                     RegAttributes... regAttributes);
285
306 template <typename... RegAttributes>
307 static __forceinline__ __device__ void optixMakeHitObjectWithRecord(OptixTraversableHandle        handle,
308                                                                 float3                          rayOrigin,
309                                                                 float3                          rayDirection,
310                                                                       float                           tmin,
311                                                                       float                           tmax,
312                                                                     float                         rayTime,
313                                                                 unsigned int                  sbtRecordIndex,
314                                                                       unsigned int                  instIdx,
315                                                                   const OptixTraversableHandle* transforms,
316                                                                 unsigned int                  numTransforms,
317                                                                    unsigned int                  sbtGASIdx,
318                                                                      unsigned int                  primIdx,
319                                                                      unsigned int                  hitKind,
320                                                                        RegAttributes... regAttributes);
321
334 static __forceinline__ __device__ void optixMakeMissHitObject(unsigned int missSBTIndex,
335                                                           float3       rayOrigin,
336                                                           float3       rayDirection,
337                                                           float        tmin,
338                                                           float        tmax,
339                                                           float        rayTime);
340
348 static __forceinline__ __device__ void optixMakeNopHitObject();
349
353 static __forceinline__ __device__ bool optixHitObjectIsHit();
354
358 static __forceinline__ __device__ bool optixHitObjectIsMiss();
359
365 static __forceinline__ __device__ bool optixHitObjectIsNop();
366
373 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex();
374
380 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p);
381 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p);
382 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p);
383 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p);
384 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p);
385 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p);
386 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p);
387 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p);
388 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p);
389 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p);
390 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p);
391 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p);
392 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p);
```

```
393 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p);
394 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p);
395 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p);
396 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p);
397 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p);
398 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p);
399 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p);
400 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p);
401 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p);
402 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p);
403 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p);
404 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p);
405 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p);
406 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p);
407 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p);
408 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p);
409 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p);
410 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p);
411 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p);
412
418 static __forceinline__ __device__ unsigned int optixGetPayload_0();
419 static __forceinline__ __device__ unsigned int optixGetPayload_1();
420 static __forceinline__ __device__ unsigned int optixGetPayload_2();
421 static __forceinline__ __device__ unsigned int optixGetPayload_3();
422 static __forceinline__ __device__ unsigned int optixGetPayload_4();
423 static __forceinline__ __device__ unsigned int optixGetPayload_5();
424 static __forceinline__ __device__ unsigned int optixGetPayload_6();
425 static __forceinline__ __device__ unsigned int optixGetPayload_7();
426 static __forceinline__ __device__ unsigned int optixGetPayload_8();
427 static __forceinline__ __device__ unsigned int optixGetPayload_9();
428 static __forceinline__ __device__ unsigned int optixGetPayload_10();
429 static __forceinline__ __device__ unsigned int optixGetPayload_11();
430 static __forceinline__ __device__ unsigned int optixGetPayload_12();
431 static __forceinline__ __device__ unsigned int optixGetPayload_13();
432 static __forceinline__ __device__ unsigned int optixGetPayload_14();
433 static __forceinline__ __device__ unsigned int optixGetPayload_15();
434 static __forceinline__ __device__ unsigned int optixGetPayload_16();
435 static __forceinline__ __device__ unsigned int optixGetPayload_17();
436 static __forceinline__ __device__ unsigned int optixGetPayload_18();
437 static __forceinline__ __device__ unsigned int optixGetPayload_19();
438 static __forceinline__ __device__ unsigned int optixGetPayload_20();
439 static __forceinline__ __device__ unsigned int optixGetPayload_21();
440 static __forceinline__ __device__ unsigned int optixGetPayload_22();
441 static __forceinline__ __device__ unsigned int optixGetPayload_23();
442 static __forceinline__ __device__ unsigned int optixGetPayload_24();
443 static __forceinline__ __device__ unsigned int optixGetPayload_25();
444 static __forceinline__ __device__ unsigned int optixGetPayload_26();
445 static __forceinline__ __device__ unsigned int optixGetPayload_27();
446 static __forceinline__ __device__ unsigned int optixGetPayload_28();
447 static __forceinline__ __device__ unsigned int optixGetPayload_29();
448 static __forceinline__ __device__ unsigned int optixGetPayload_30();
449 static __forceinline__ __device__ unsigned int optixGetPayload_31();
450
459 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int typeMask);
460
464 static __forceinline__ __device__ unsigned int optixUndefinedValue();
465
472 static __forceinline__ __device__ float3 optixGetWorldRayOrigin();
473
480 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin();
481
488 static __forceinline__ __device__ float3 optixGetWorldRayDirection();
489
496 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection();
497
501 static __forceinline__ __device__ float3 optixGetObjectRayOrigin();
502
```

```
506 static __forceinline__ __device__ float3 optixGetObjectRayDirection();
507
511 static __forceinline__ __device__ float optixGetRayTmin();
512
519 static __forceinline__ __device__ float optixHitObjectGetRayTmin();
520
529 static __forceinline__ __device__ float optixGetRayTmax();
530
539 static __forceinline__ __device__ float optixHitObjectGetRayTmax();
540
546 static __forceinline__ __device__ float optixGetRayTime();
547
554 static __forceinline__ __device__ float optixHitObjectGetRayTime();
555
559 static __forceinline__ __device__ unsigned int optixGetRayFlags();
560
564 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask();
565
572 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias, unsigned int instIdx);
573
584 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas, unsigned
int primIdx, unsigned int sbtGASIndex, float time, float3 data[3]);
585
590 static __forceinline__ __device__ void optixGetMicroTriangleVertexData(float3 data[3]);
591
596 static __forceinline__ __device__ void optixGetMicroTriangleBarycentricsData(float2 data[3]);
597
610 static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2]);
611
624 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3]);
625
638 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4]);
639
652 static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas, unsigned
int primIdx, unsigned int sbtGASIndex, float time, float4 data[4]);
653
666 static __forceinline__ __device__ void optixGetCubicBezierVertexData(OptixTraversableHandle gas,
unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4]);
667
680 static __forceinline__ __device__ void optixGetRibbonVertexData(OptixTraversableHandle gas, unsigned int
primIdx, unsigned int sbtGASIndex, float time, float4 data[3]);
681
685 static __forceinline__ __device__ float3 optixGetRibbonNormal(OptixTraversableHandle gas, unsigned int
primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters);
686
699 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas, unsigned int
primIdx, unsigned int sbtGASIndex, float time, float4 data[1]);
700
705 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle();
706
710 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle gas);
711
715 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle gas);
716
720 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle gas);
721
728 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12]);
729
736 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12]);
737
744 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point);
745
752 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec);
```

```
753
760 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal);
761
768 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point);
769
776 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec);
777
784 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal);
785
789 static __forceinline__ __device__ unsigned int optixGetTransformListSize();
790
799 static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize();
800
804 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int index);
805
814 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle(unsigned
int index);
815
819 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle);
820
826 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle);
827
833 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle);
834
840 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle);
841
847 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle);
848
854 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle(OptixTraversableHandle handle);
855
861 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle);
862
868 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle);
869
893 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind);
894
900 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0);
901
907 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1);
908
914 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1, unsigned int a2);
915
921 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
922                                                                unsigned int hitKind,
923                                                                unsigned int a0,
924                                                                unsigned int a1,
925                                                                unsigned int a2,
926                                                                unsigned int a3);
927
933 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
934                                                                unsigned int hitKind,
935                                                                unsigned int a0,
936                                                                unsigned int a1,
937                                                                unsigned int a2,
938                                                                unsigned int a3,
939                                                                unsigned int a4);
```

```
940
946 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
947                                                                 unsigned int hitKind,
948                                                                 unsigned int a0,
949                                                                 unsigned int a1,
950                                                                 unsigned int a2,
951                                                                 unsigned int a3,
952                                                                 unsigned int a4,
953                                                                 unsigned int a5);
954
960 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
961                                                                 unsigned int hitKind,
962                                                                 unsigned int a0,
963                                                                 unsigned int a1,
964                                                                 unsigned int a2,
965                                                                 unsigned int a3,
966                                                                 unsigned int a4,
967                                                                 unsigned int a5,
968                                                                 unsigned int a6);
969
975 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
976                                                                 unsigned int hitKind,
977                                                                 unsigned int a0,
978                                                                 unsigned int a1,
979                                                                 unsigned int a2,
980                                                                 unsigned int a3,
981                                                                 unsigned int a4,
982                                                                 unsigned int a5,
983                                                                 unsigned int a6,
984                                                                 unsigned int a7);
985
990 static __forceinline__ __device__ unsigned int optixGetAttribute_0();
991 static __forceinline__ __device__ unsigned int optixGetAttribute_1();
992 static __forceinline__ __device__ unsigned int optixGetAttribute_2();
993 static __forceinline__ __device__ unsigned int optixGetAttribute_3();
994 static __forceinline__ __device__ unsigned int optixGetAttribute_4();
995 static __forceinline__ __device__ unsigned int optixGetAttribute_5();
996 static __forceinline__ __device__ unsigned int optixGetAttribute_6();
997 static __forceinline__ __device__ unsigned int optixGetAttribute_7();
998
999
1007 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0();
1008 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1();
1009 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2();
1010 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3();
1011 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4();
1012 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5();
1013 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6();
1014 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7();
1015
1019 static __forceinline__ __device__ void optixTerminateRay();
1020
1025 static __forceinline__ __device__ void optixIgnoreIntersection();
1026
1027
1043 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex();
1044
1052 static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex();
1053
1064 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex();
1065
1074 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex();
1075
1076
1089 static __forceinline__ __device__ unsigned int optixGetInstanceId();
1090
1099 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId();
```

```
1100
1110 static __forceinline__ __device__ unsigned int optixGetInstanceIndex();
1111
1120 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex();
1121
1129 static __forceinline__ __device__ unsigned int optixGetHitKind();
1130
1138 static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind();
1139
1143 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind);
1144
1148 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind);
1149
1153 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind);
1154
1158 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType();
1159
1163 static __forceinline__ __device__ bool optixIsFrontFaceHit();
1164
1168 static __forceinline__ __device__ bool optixIsBackFaceHit();
1169
1173 static __forceinline__ __device__ bool optixIsTriangleHit();
1174
1178 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit();
1179
1183 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit();
1184
1188 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleHit();
1189
1193 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleFrontFaceHit();
1194
1198 static __forceinline__ __device__ bool optixIsDisplacedMicromeshTriangleBackFaceHit();
1199
1206 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics();
1207
1212 static __forceinline__ __device__ float optixGetCurveParameter();
1213
1219 static __forceinline__ __device__ float2 optixGetRibbonParameters();
1220
1227 static __forceinline__ __device__ uint3 optixGetLaunchIndex();
1228
1233 static __forceinline__ __device__ uint3 optixGetLaunchDimensions();
1234
1242 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer();
1243
1250 static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer();
1251
1266 static __forceinline__ __device__ void optixThrowException(int exceptionCode);
1267
1273 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0);
1274
1280 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1281                                                            unsigned int exceptionDetail0,
1282                                                            unsigned int exceptionDetail1);
1283
1289 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1290                                                            unsigned int exceptionDetail0,
1291                                                            unsigned int exceptionDetail1,
1292                                                            unsigned int exceptionDetail2);
1293
1299 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1300                                                            unsigned int exceptionDetail0,
1301                                                            unsigned int exceptionDetail1,
1302                                                            unsigned int exceptionDetail2,
1303                                                            unsigned int exceptionDetail3);
1304
```

```
1310 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1311                                                             unsigned int exceptionDetail0,
1312                                                             unsigned int exceptionDetail1,
1313                                                             unsigned int exceptionDetail2,
1314                                                             unsigned int exceptionDetail3,
1315                                                             unsigned int exceptionDetail4);
1316
1322 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1323                                                             unsigned int exceptionDetail0,
1324                                                             unsigned int exceptionDetail1,
1325                                                             unsigned int exceptionDetail2,
1326                                                             unsigned int exceptionDetail3,
1327                                                             unsigned int exceptionDetail4,
1328                                                             unsigned int exceptionDetail5);
1329
1336 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1337                                                             unsigned int exceptionDetail0,
1338                                                             unsigned int exceptionDetail1,
1339                                                             unsigned int exceptionDetail2,
1340                                                             unsigned int exceptionDetail3,
1341                                                             unsigned int exceptionDetail4,
1342                                                             unsigned int exceptionDetail5,
1343                                                             unsigned int exceptionDetail6);
1344
1350 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1351                                                             unsigned int exceptionDetail0,
1352                                                             unsigned int exceptionDetail1,
1353                                                             unsigned int exceptionDetail2,
1354                                                             unsigned int exceptionDetail3,
1355                                                             unsigned int exceptionDetail4,
1356                                                             unsigned int exceptionDetail5,
1357                                                             unsigned int exceptionDetail6,
1358                                                             unsigned int exceptionDetail7);
1359
1363 static __forceinline__ __device__ int optixGetExceptionCode();
1364
1371 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0();
1372
1378 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1();
1379
1385 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2();
1386
1392 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3();
1393
1399 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4();
1400
1406 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5();
1407
1413 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6();
1414
1420 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7();
1421
1422
1435 static __forceinline__ __device__ char* optixGetExceptionLineInfo();
1436
1460 template <typename ReturnT, typename... ArgTypes>
1461 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args);
1462
1463
1486 template <typename ReturnT, typename... ArgTypes>
1487 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes...
args);
1488
1489
1554 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int
texInfo, float x, float y, unsigned int* singleMipLevel);
1555
```

```
1567 static __forceinline__ __device__ uint4
1568 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool
coarse, unsigned int* singleMipLevel);
1569
1584 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
1585                                                                 unsigned int       texInfo,
1586                                                                 float              x,
1587                                                                 float              y,
1588                                                                 float              dPdx_x,
1589                                                                 float              dPdx_y,
1590                                                                 float              dPdy_x,
1591                                                                 float              dPdy_y,
1592                                                                 bool               coarse,
1593                                                                 unsigned int*      singleMipLevel);
1594   // end group optix_device_api
1596
1597 #define __OPTIX_INCLUDE_INTERNAL_HEADERS__
1598
1599 #include "internal/optix_device_impl.h"
1600
1601 #endif  // OPTIX_OPTIX_DEVICE_H
```

8.13 optix_function_table.h File Reference

Classes

- struct OptixFunctionTable

Macros

- #define OPTIX_ABI_VERSION 87

Typedefs

- typedef struct OptixFunctionTable OptixFunctionTable

8.13.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.13.2 Macro Definition Documentation

8.13.2.1 OPTIX_ABI_VERSION

```
#define OPTIX_ABI_VERSION 87
```

The OptiX ABI version.

8.14 optix_function_table.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2023 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 87
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 (*optixModuleCreate)(OptixDeviceContext                 context,
109                                      const OptixModuleCompileOptions*   moduleCompileOptions,
110                                      const OptixPipelineCompileOptions* pipelineCompileOptions,
111                                      const char*                        input,
112                                      size_t                             inputSize,
113                                      char*                              logString,
114                                      size_t*                            logStringSize,
115                                      OptixModule*                       module);
116
118     OptixResult (*optixModuleCreateWithTasks)(OptixDeviceContext                 context,
119                                               const OptixModuleCompileOptions*   moduleCompileOptions,
120                                               const OptixPipelineCompileOptions* pipelineCompileOptions,
121                                               const char*                        input,
122                                               size_t                             inputSize,
123                                               char*                              logString,
124                                               size_t*                            logStringSize,
125                                               OptixModule*                       module,
126                                               OptixTask*                         firstTask);
127
129     OptixResult (*optixModuleGetCompilationState)(OptixModule module, OptixModuleCompileState* state);
130
132     OptixResult (*optixModuleDestroy)(OptixModule module);
133
135     OptixResult(*optixBuiltinISModuleGet)(OptixDeviceContext                 context,
136                                           const OptixModuleCompileOptions*   moduleCompileOptions,
137                                           const OptixPipelineCompileOptions* pipelineCompileOptions,
138                                           const OptixBuiltinISOptions*       builtinISOptions,
139                                           OptixModule*                       builtinModule);
140
141     //@ }
143     //@ {
144
146     OptixResult (*optixTaskExecute)(OptixTask     task,
147                                     OptixTask*    additionalTasks,
148                                     unsigned int  maxNumAdditionalTasks,
149                                     unsigned int* numAdditionalTasksCreated);
150     //@ }
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, OptixPipeline pipeline);
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
246     OptixResult (*optixAccelEmitProperty)(OptixDeviceContext        context,
247                                           CUstream                  stream,
248                                           OptixTraversableHandle    handle,
249                                           const OptixAccelEmitDesc* emittedProperty);
250
252     OptixResult (*optixConvertPointerToTraversableHandle)(OptixDeviceContext      onDevice,
```

```
253                                                         CUdeviceptr             pointer,
254                                                         OptixTraversableType    traversableType,
255                                                         OptixTraversableHandle* traversableHandle);
256
258     OptixResult (*optixOpacityMicromapArrayComputeMemoryUsage)(OptixDeviceContext
context,
259                                                           const OptixOpacityMicromapArrayBuildInput*
buildInput,
260                                                              OptixMicromapBufferSizes*
bufferSizes);
261
263     OptixResult (*optixOpacityMicromapArrayBuild)(OptixDeviceContext                        context,
264                                                     CUstream                                  stream,
265                                                 const OptixOpacityMicromapArrayBuildInput* buildInput,
266                                                     const OptixMicromapBuffers*               buffers);
267
269     OptixResult (*optixOpacityMicromapArrayGetRelocationInfo)(OptixDeviceContext   context,
270                                                           CUdeviceptr         opacityMicromapArray,
271                                                             OptixRelocationInfo* info);
272
274     OptixResult (*optixOpacityMicromapArrayRelocate)(OptixDeviceContext         context,
275                                                         CUstream                  stream,
276                                                         const OptixRelocationInfo* info,
277                                                     CUdeviceptr               targetOpacityMicromapArray,
278                                                         size_t
targetOpacityMicromapArraySizeInBytes);
279
281     OptixResult (*optixDisplacementMicromapArrayComputeMemoryUsage)(OptixDeviceContext context,
282                                                                     const
OptixDisplacementMicromapArrayBuildInput* buildInput,
283                                                                  OptixMicromapBufferSizes* bufferSizes);
284
286     OptixResult (*optixDisplacementMicromapArrayBuild)(OptixDeviceContext
context,
287                                                     CUstream                                  stream,
288                                                         const OptixDisplacementMicromapArrayBuildInput*
buildInput,
289                                                          const OptixMicromapBuffers*
buffers);
290
291     //@ }
293     //@ {
294
296     OptixResult (*optixSbtRecordPackHeader)(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer);
297
299     OptixResult (*optixLaunch)(OptixPipeline                 pipeline,
300                                CUstream                      stream,
301                                CUdeviceptr                   pipelineParams,
302                                size_t                        pipelineParamsSize,
303                                const OptixShaderBindingTable* sbt,
304                                unsigned int                  width,
305                                unsigned int                  height,
306                                unsigned int                  depth);
307
308     //@ }
310     //@ {
311
313     OptixResult (*optixDenoiserCreate)(OptixDeviceContext context, OptixDenoiserModelKind modelKind,
const OptixDenoiserOptions* options, OptixDenoiser* returnHandle);
314
316     OptixResult (*optixDenoiserDestroy)(OptixDenoiser handle);
317
319     OptixResult (*optixDenoiserComputeMemoryResources)(const OptixDenoiser handle,
320                                                        unsigned int        maximumInputWidth,
321                                                        unsigned int        maximumInputHeight,
322                                                        OptixDenoiserSizes* returnSizes);
```

```
323
325     OptixResult (*optixDenoiserSetup)(OptixDenoiser denoiser,
326                                       CUstream      stream,
327                                       unsigned int  inputWidth,
328                                       unsigned int  inputHeight,
329                                       CUdeviceptr   state,
330                                       size_t        stateSizeInBytes,
331                                       CUdeviceptr   scratch,
332                                       size_t        scratchSizeInBytes);
333
335     OptixResult (*optixDenoiserInvoke)(OptixDenoiser                   denoiser,
336                                        CUstream                        stream,
337                                        const OptixDenoiserParams*      params,
338                                        CUdeviceptr                     denoiserState,
339                                        size_t                          denoiserStateSizeInBytes,
340                                        const OptixDenoiserGuideLayer * guideLayer,
341                                        const OptixDenoiserLayer *      layers,
342                                        unsigned int                    numLayers,
343                                        unsigned int                    inputOffsetX,
344                                        unsigned int                    inputOffsetY,
345                                        CUdeviceptr                     scratch,
346                                        size_t                          scratchSizeInBytes);
347
349     OptixResult (*optixDenoiserComputeIntensity)(OptixDenoiser       handle,
350                                                  CUstream            stream,
351                                                  const OptixImage2D* inputImage,
352                                                  CUdeviceptr         outputIntensity,
353                                                  CUdeviceptr         scratch,
354                                                  size_t              scratchSizeInBytes);
355
357     OptixResult (*optixDenoiserComputeAverageColor)(OptixDenoiser       handle,
358                                                     CUstream            stream,
359                                                     const OptixImage2D* inputImage,
360                                                     CUdeviceptr         outputAverageColor,
361                                                     CUdeviceptr         scratch,
362                                                     size_t              scratchSizeInBytes);
363
365     OptixResult (*optixDenoiserCreateWithUserModel)(OptixDeviceContext context, const void * data, size_t
dataSizeInBytes, OptixDenoiser* returnHandle);
366     //@ }
367
368 } OptixFunctionTable;
369   // end group optix_function_table
371
372 #ifdef __cplusplus
373 }
374 #endif
375
376 #endif /* OPTIX_DEFINE_ABI_VERSION_ONLY */
377
378 #endif /* OPTIX_OPTIX_FUNCTION_TABLE_H */
```

8.15 optix_function_table_definition.h File Reference

Variables

- OptixFunctionTable g_optixFunctionTable

8.15.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.16 optix_function_table_definition.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2023 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_FUNCTION_TABLE_DEFINITION_H
34 #define OPTIX_OPTIX_FUNCTION_TABLE_DEFINITION_H
35
36 #include "optix_function_table.h"
37
38 #ifdef __cplusplus
39 extern "C" {
40 #endif
41
49 OptixFunctionTable g_optixFunctionTable;
50   // end group optix_function_table
52
53 #ifdef __cplusplus
54 }
55 #endif
56
57 #endif  // OPTIX_OPTIX_FUNCTION_TABLE_DEFINITION_H
```

8.17 optix_host.h File Reference

Functions

- const char * optixGetErrorName (OptixResult result)
- const char * optixGetErrorString (OptixResult result)
- OptixResult optixDeviceContextCreate (CUcontext fromContext, const OptixDeviceContextOptions *options, OptixDeviceContext *context)
- OptixResult optixDeviceContextDestroy (OptixDeviceContext context)

- OptixResult optixDeviceContextGetProperty (OptixDeviceContext context, OptixDeviceProperty property, void *value, size_t sizeInBytes)
- OptixResult optixDeviceContextSetLogCallback (OptixDeviceContext context, OptixLogCallback callbackFunction, void *callbackData, unsigned int callbackLevel)
- OptixResult optixDeviceContextSetCacheEnabled (OptixDeviceContext context, int enabled)
- OptixResult optixDeviceContextSetCacheLocation (OptixDeviceContext context, const char *location)
- OptixResult optixDeviceContextSetCacheDatabaseSizes (OptixDeviceContext context, size_t lowWaterMark, size_t highWaterMark)
- OptixResult optixDeviceContextGetCacheEnabled (OptixDeviceContext context, int *enabled)
- OptixResult optixDeviceContextGetCacheLocation (OptixDeviceContext context, char *location, size_t locationSize)
- OptixResult optixDeviceContextGetCacheDatabaseSizes (OptixDeviceContext context, size_t *lowWaterMark, size_t *highWaterMark)
- OptixResult optixPipelineCreate (OptixDeviceContext context, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixPipelineLinkOptions *pipelineLinkOptions, const OptixProgramGroup *programGroups, unsigned int numProgramGroups, char *logString, size_t *logStringSize, OptixPipeline *pipeline)
- OptixResult optixPipelineDestroy (OptixPipeline pipeline)
- OptixResult optixPipelineSetStackSize (OptixPipeline pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)
- OptixResult optixModuleCreate (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module)
- OptixResult optixModuleCreateWithTasks (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module, OptixTask *firstTask)
- OptixResult optixModuleGetCompilationState (OptixModule module, OptixModuleCompileState *state)
- OptixResult optixModuleDestroy (OptixModule module)
- OptixResult optixBuiltinISModuleGet (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixBuiltinISOptions *builtinISOptions, OptixModule *builtinModule)
- OptixResult optixTaskExecute (OptixTask task, OptixTask *additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int *numAdditionalTasksCreated)
- OptixResult optixProgramGroupGetStackSize (OptixProgramGroup programGroup, OptixStackSizes *stackSizes, OptixPipeline pipeline)
- OptixResult optixProgramGroupCreate (OptixDeviceContext context, const OptixProgramGroupDesc *programDescriptions, unsigned int numProgramGroups, const OptixProgramGroupOptions *options, char *logString, size_t *logStringSize, OptixProgramGroup *programGroups)
- OptixResult optixProgramGroupDestroy (OptixProgramGroup programGroup)
- OptixResult optixLaunch (OptixPipeline pipeline, CUstream stream, CUdeviceptr pipelineParams, size_t pipelineParamsSize, const OptixShaderBindingTable *sbt, unsigned int width, unsigned int height, unsigned int depth)
- OptixResult optixSbtRecordPackHeader (OptixProgramGroup programGroup, void *sbtRecordHeaderHostPointer)

- OptixResult optixAccelComputeMemoryUsage (OptixDeviceContext context, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, OptixAccelBufferSizes *bufferSizes)
- OptixResult optixAccelBuild (OptixDeviceContext context, CUstream stream, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, CUdeviceptr tempBuffer, size_t tempBufferSizeInBytes, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle, const OptixAccelEmitDesc *emittedProperties, unsigned int numEmittedProperties)
- OptixResult optixAccelGetRelocationInfo (OptixDeviceContext context, OptixTraversableHandle handle, OptixRelocationInfo *info)
- OptixResult optixCheckRelocationCompatibility (OptixDeviceContext context, const OptixRelocationInfo *info, int *compatible)
- OptixResult optixAccelRelocate (OptixDeviceContext context, CUstream stream, const OptixRelocationInfo *info, const OptixRelocateInput *relocateInputs, size_t numRelocateInputs, CUdeviceptr targetAccel, size_t targetAccelSizeInBytes, OptixTraversableHandle *targetHandle)
- OptixResult optixAccelCompact (OptixDeviceContext context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle)
- OptixResult optixAccelEmitProperty (OptixDeviceContext context, CUstream stream, OptixTraversableHandle handle, const OptixAccelEmitDesc *emittedProperty)
- OptixResult optixConvertPointerToTraversableHandle (OptixDeviceContext onDevice, CUdeviceptr pointer, OptixTraversableType traversableType, OptixTraversableHandle *traversableHandle)
- OptixResult optixOpacityMicromapArrayComputeMemoryUsage (OptixDeviceContext context, const OptixOpacityMicromapArrayBuildInput *buildInput, OptixMicromapBufferSizes *bufferSizes)
- OptixResult optixOpacityMicromapArrayBuild (OptixDeviceContext context, CUstream stream, const OptixOpacityMicromapArrayBuildInput *buildInput, const OptixMicromapBuffers *buffers)
- OptixResult optixOpacityMicromapArrayGetRelocationInfo (OptixDeviceContext context, CUdeviceptr opacityMicromapArray, OptixRelocationInfo *info)
- OptixResult optixOpacityMicromapArrayRelocate (OptixDeviceContext context, CUstream stream, const OptixRelocationInfo *info, CUdeviceptr targetOpacityMicromapArray, size_t targetOpacityMicromapArraySizeInBytes)
- OptixResult optixDisplacementMicromapArrayComputeMemoryUsage (OptixDeviceContext context, const OptixDisplacementMicromapArrayBuildInput *buildInput, OptixMicromapBufferSizes *bufferSizes)
- OptixResult optixDisplacementMicromapArrayBuild (OptixDeviceContext context, CUstream stream, const OptixDisplacementMicromapArrayBuildInput *buildInput, const OptixMicromapBuffers *buffers)
- OptixResult optixDenoiserCreate (OptixDeviceContext context, OptixDenoiserModelKind modelKind, const OptixDenoiserOptions *options, OptixDenoiser *denoiser)
- OptixResult optixDenoiserCreateWithUserModel (OptixDeviceContext context, const void *userData, size_t userDataSizeInBytes, OptixDenoiser *denoiser)
- OptixResult optixDenoiserDestroy (OptixDenoiser denoiser)
- OptixResult optixDenoiserComputeMemoryResources (const OptixDenoiser denoiser, unsigned int outputWidth, unsigned int outputHeight, OptixDenoiserSizes *returnSizes)
- OptixResult optixDenoiserSetup (OptixDenoiser denoiser, CUstream stream, unsigned int inputWidth, unsigned int inputHeight, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, CUdeviceptr scratch, size_t scratchSizeInBytes)

- OptixResult optixDenoiserInvoke (OptixDenoiser denoiser, CUstream stream, const OptixDenoiserParams *params, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, const OptixDenoiserGuideLayer *guideLayer, const OptixDenoiserLayer *layers, unsigned int numLayers, unsigned int inputOffsetX, unsigned int inputOffsetY, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OptixResult optixDenoiserComputeIntensity (OptixDenoiser denoiser, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputIntensity, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OptixResult optixDenoiserComputeAverageColor (OptixDenoiser denoiser, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputAverageColor, CUdeviceptr scratch, size_t scratchSizeInBytes)

8.17.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

OptiX host include file – includes the host api if compiling host code. For the math library routines include optix_math.h

8.17.2 Function Documentation

8.17.2.1 optixAccelBuild()

```
OptixResult optixAccelBuild (
            OptixDeviceContext context,
            CUstream stream,
            const OptixAccelBuildOptions * accelOptions,
            const OptixBuildInput * buildInputs,
            unsigned int numBuildInputs,
            CUdeviceptr tempBuffer,
            size_t tempBufferSizeInBytes,
            CUdeviceptr outputBuffer,
            size_t outputBufferSizeInBytes,
            OptixTraversableHandle * outputHandle,
            const OptixAccelEmitDesc * emittedProperties,
            unsigned int numEmittedProperties )
```

Parameters

in	*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

Parameters

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)

8.17.2.2 optixAccelCompact()

```
OptixResult optixAccelCompact (
          OptixDeviceContext context,
          CUstream stream,
          OptixTraversableHandle inputHandle,
          CUdeviceptr outputBuffer,
          size_t outputBufferSizeInBytes,
          OptixTraversableHandle * outputHandle )
```

After building an acceleration structure, it can be copied in a compacted form to reduce memory. In order to be compacted, OPTIX_BUILD_FLAG_ALLOW_COMPACTION must be supplied in OptixAccelBuildOptions::buildFlags passed to optixAccelBuild.

'outputBuffer' is the pointer to where the compacted acceleration structure will be written. This pointer must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT.

The size of the memory specified in 'outputBufferSizeInBytes' should be at least the value computed using the OPTIX_PROPERTY_TYPE_COMPACTED_SIZE that was reported during optixAccelBuild.

Parameters

in	*context*
in	*stream*
in	*inputHandle*
in	*outputBuffer*
in	*outputBufferSizeInBytes*
out	*outputHandle*

8.17.2.3 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

8.17.2.4 optixAccelEmitProperty()

```
OptixResult optixAccelEmitProperty (
            OptixDeviceContext context,
            CUstream stream,
            OptixTraversableHandle handle,
            const OptixAccelEmitDesc * emittedProperty )
```

Emit a single property after an acceleration structure was built. The result buffer of the ' emittedProperty' needs to be large enough to hold the requested property (.

See also OptixAccelPropertyType).

Parameters

in	*context*	
in	*stream*	
in	*handle*	
in	*emittedProperty*	type of requested property and output buffer

8.17.2.5 optixAccelGetRelocationInfo()

```
OptixResult optixAccelGetRelocationInfo (
            OptixDeviceContext context,
            OptixTraversableHandle handle,
            OptixRelocationInfo * info )
```

Obtain relocation information, stored in OptixRelocationInfo, for a given context and acceleration structure's traversable handle.

The relocation information can be passed to optixCheckRelocationCompatibility to determine if an acceleration structure, referenced by 'handle', can be relocated to a different device's memory space (see optixCheckRelocationCompatibility).

When used with optixAccelRelocate, it provides data necessary for doing the relocation.

If the acceleration structure data associated with 'handle' is copied multiple times, the same OptixRelocationInfo can also be used on all copies.

Parameters

in	*context*
in	*handle*

Parameters

out	*info*

Returns

OPTIX_ERROR_INVALID_VALUE will be returned for traversable handles that are not from acceleration structure builds.

8.17.2.6 optixAccelRelocate()

```
OptixResult optixAccelRelocate (
            OptixDeviceContext context,
            CUstream stream,
            const OptixRelocationInfo * info,
            const OptixRelocateInput * relocateInputs,
            size_t numRelocateInputs,
            CUdeviceptr targetAccel,
            size_t targetAccelSizeInBytes,
            OptixTraversableHandle * targetHandle )
```

optixAccelRelocate is called to update the acceleration structure after it has been relocated. Relocation is necessary when the acceleration structure's location in device memory has changed. optixAccelRelocate does not copy the memory. This function only operates on the relocated memory whose new location is specified by 'targetAccel'. optixAccelRelocate also returns the new OptixTraversableHandle associated with 'targetAccel'. The original memory (source) is not required to be valid, only the OptixRelocationInfo.

Before calling optixAccelRelocate, optixCheckRelocationCompatibility should be called to ensure the copy will be compatible with the destination device context.

The memory pointed to by 'targetAccel' should be allocated with the same size as the source acceleration. Similar to the 'outputBuffer' used in optixAccelBuild, this pointer must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT.

The memory in 'targetAccel' must be allocated as long as the accel is in use.

The instance traversables referenced by an IAS and the micromaps referenced by a triangle GAS may themselves require relocation. 'relocateInputs' and 'numRelocateInputs' should be used to specify the relocated traversables and micromaps. After relocation, the relocated accel will reference these relocated traversables and micromaps instead of their sources. The number of relocate inputs 'numRelocateInputs' must match the number of build inputs 'numBuildInputs' used to build the source accel. Relocation inputs correspond with build inputs used to build the source accel and should appear in the same order (see optixAccelBuild). 'relocateInputs' and 'numRelocateInputs' may be zero, preserving any references to traversables and micromaps from the source accel.

Parameters

in	*context*
in	*stream*
in	*info*
in	*relocateInputs*

Parameters

in	*numRelocateInputs*
in	*targetAccel*
in	*targetAccelSizeInBytes*
out	*targetHandle*

8.17.2.7 optixBuiltinISModuleGet()

```
OptixResult optixBuiltinISModuleGet (
            OptixDeviceContext context,
            const OptixModuleCompileOptions * moduleCompileOptions,
            const OptixPipelineCompileOptions * pipelineCompileOptions,
            const OptixBuiltinISOptions * builtinISOptions,
            OptixModule * builtinModule )
```

Returns a module containing the intersection program for the built-in primitive type specified by the builtinISOptions. This module must be used as the moduleIS for the OptixProgramGroupHitgroup in any SBT record for that primitive type. (The entryFunctionNameIS should be null.)

8.17.2.8 optixCheckRelocationCompatibility()

```
OptixResult optixCheckRelocationCompatibility (
            OptixDeviceContext context,
            const OptixRelocationInfo * info,
            int * compatible )
```

Checks if an optix data structure built using another OptixDeviceContext (that was used to fill in 'info') is compatible with the OptixDeviceContext specified in the 'context' parameter.

Any device is always compatible with itself.

Parameters

in	*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.

8.17.2.9 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

8.17.2.10 optixDenoiserComputeAverageColor()

```
OptixResult optixDenoiserComputeAverageColor (
            OptixDenoiser denoiser,
            CUstream stream,
            const OptixImage2D * inputImage,
            CUdeviceptr outputAverageColor,
            CUdeviceptr scratch,
            size_t scratchSizeInBytes )
```

Compute average logarithmic for each of the first three channels for the given image. When denoising tiles the intensity of the entire image should be computed, i.e. not per tile to get consistent results.

The size of scratch memory required can be queried with optixDenoiserComputeMemoryResources.

data type unsigned char is not supported for 'inputImage', it must be 3 or 4 component half/float.

Parameters

in	*denoiser*	
in	*stream*	
in	*inputImage*	
out	*outputAverageColor*	three floats
in	*scratch*	
in	*scratchSizeInBytes*	

8.17.2.11 optixDenoiserComputeIntensity()

```
OptixResult optixDenoiserComputeIntensity (
            OptixDenoiser denoiser,
            CUstream stream,
            const OptixImage2D * inputImage,
            CUdeviceptr outputIntensity,
            CUdeviceptr scratch,
            size_t scratchSizeInBytes )
```

Computes the logarithmic average intensity of the given image. The returned value 'outputIntensity' is multiplied with the RGB values of the input image/tile in optixDenoiserInvoke if given in the parameter OptixDenoiserParams::hdrIntensity (otherwise 'hdrIntensity' must be a null pointer). This is useful for denoising HDR images which are very dark or bright. When denoising tiles the intensity of

the entire image should be computed, i.e. not per tile to get consistent results.

For each RGB pixel in the inputImage the intensity is calculated and summed if it is greater than 1e-8f: intensity = log(r ∗ 0.212586f + g ∗ 0.715170f + b ∗ 0.072200f). The function returns 0.18 / exp(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*	

8.17.2.12 optixDenoiserComputeMemoryResources()

```
OptixResult optixDenoiserComputeMemoryResources (
            const OptixDenoiser denoiser,
            unsigned int outputWidth,
            unsigned int outputHeight,
            OptixDenoiserSizes * returnSizes )
```

Computes the GPU memory resources required to execute the denoiser.

Memory for state and scratch buffers must be allocated with the sizes in 'returnSizes' and scratch memory passed to optixDenoiserSetup, optixDenoiserInvoke, optixDenoiserComputeIntensity and optixDenoiserComputeAverageColor. For tiled denoising an overlap area ('overlapWindowSizeInPixels') must be added to each tile on all sides which increases the amount of memory needed to denoise a tile. In case of tiling use withOverlapScratchSizeInBytes for scratch memory size. If only full resolution images are denoised, withoutOverlapScratchSizeInBytes can be used which is always smaller than withOverlapScratchSizeInBytes.

'outputWidth' and 'outputHeight' is the dimension of the image to be denoised (without overlap in case tiling is being used). 'outputWidth' and 'outputHeight' must be greater than or equal to the dimensions passed to optixDenoiserSetup.

Parameters

in	*denoiser*
in	*outputWidth*
in	*outputHeight*
out	*returnSizes*

8.17.2.13 optixDenoiserCreate()

```
OptixResult optixDenoiserCreate (
            OptixDeviceContext context,
            OptixDenoiserModelKind modelKind,
            const OptixDenoiserOptions * options,
            OptixDenoiser * denoiser )
```

Creates a denoiser object with the given options, using built-in inference models.

'modelKind' selects the model used for inference. Inference for the built-in models can be guided (giving hints to improve image quality) with albedo and normal vector images in the guide layer (see 'optixDenoiserInvoke'). Use of these images must be enabled in 'OptixDenoiserOptions'.

Parameters

in	*context*
in	*modelKind*
in	*options*
out	*denoiser*

8.17.2.14 optixDenoiserCreateWithUserModel()

```
OptixResult optixDenoiserCreateWithUserModel (
            OptixDeviceContext context,
            const void * userData,
            size_t userDataSizeInBytes,
            OptixDenoiser * denoiser )
```

Creates a denoiser object with the given options, using a provided inference model.

'userData' and 'userDataSizeInBytes' provide a user model for inference. The memory passed in userData will be accessed only during the invocation of this function and can be freed after it returns. The user model must export only one weight set which determines both the model kind and the required set of guide images.

Parameters

in	*context*
in	*userData*
in	*userDataSizeInBytes*
out	*denoiser*

8.17.2.15 optixDenoiserDestroy()

```
OptixResult optixDenoiserDestroy (
            OptixDenoiser denoiser )
```

Destroys the denoiser object and any associated host resources.

8.17.2.16 optixDenoiserInvoke()

```
OptixResult optixDenoiserInvoke (
            OptixDenoiser denoiser,
            CUstream stream,
            const OptixDenoiserParams * params,
            CUdeviceptr denoiserState,
            size_t denoiserStateSizeInBytes,
            const OptixDenoiserGuideLayer * guideLayer,
            const OptixDenoiserLayer * layers,
            unsigned int numLayers,
            unsigned int inputOffsetX,
            unsigned int inputOffsetY,
            CUdeviceptr scratch,
            size_t scratchSizeInBytes )
```

Invokes denoiser on a set of input data and produces at least one output image. State memory must be available during the execution of the denoiser (or until optixDenoiserSetup is called with a new state memory pointer). Scratch memory passed is used only for the duration of this function. Scratch and state memory sizes must have a size greater than or equal to the sizes as returned by optixDenoiserComputeMemoryResources.

'inputOffsetX' and 'inputOffsetY' are pixel offsets in the 'inputLayers' image specifying the beginning of the image without overlap. When denoising an entire image without tiling there is no overlap and 'inputOffsetX' and 'inputOffsetY' must be zero. When denoising a tile which is adjacent to one of the four sides of the entire image the corresponding offsets must also be zero since there is no overlap at the side adjacent to the image border.

'guideLayer' provides additional information to the denoiser. When providing albedo and normal vector guide images, the corresponding fields in the 'OptixDenoiserOptions' must be enabled, see optixDenoiserCreate. 'guideLayer' must not be null. If a guide image in 'OptixDenoiserOptions' is not enabled, the corresponding image in 'OptixDenoiserGuideLayer' is ignored.

If OPTIX_DENOISER_MODEL_KIND_TEMPORAL or OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV is selected, a 2d flow image must be given in 'OptixDenoiserGuideLayer'. It describes for each pixel the flow from the previous to the current frame (a 2d vector in pixel space). The denoised beauty/AOV of the previous frame must be given in 'previousOutput'. If this image is not available in the first frame of a sequence, the noisy beauty/AOV from the first frame and zero flow vectors could be given as a substitute. For non-temporal model kinds the flow image in 'OptixDenoiserGuideLayer' is ignored. 'previousOutput' and 'output' may refer to the same buffer if tiling is not used, i.e. 'previousOutput' is first read by this function and later overwritten with the denoised result. 'output' can be passed as 'previousOutput' to the next frame. In other model kinds (not temporal) 'previousOutput' is ignored.

The beauty layer must be given as the first entry in 'layers'. In AOV type model kinds (OPTIX_DENOISER_MODEL_KIND_AOV or in user defined models implementing kernel-prediction) additional layers for the AOV images can be given. In each layer the noisy input image is given in 'input', the denoised output is written into the 'output' image. input and output images may refer to the same buffer, with the restriction that the pixel formats must be identical for input and output when the blend mode is selected (see OptixDenoiserParams).

If OPTIX_DENOISER_MODEL_KIND_TEMPORAL or OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV is selected, the denoised image from the previous frame must be given in 'previousOutput' in the layer. 'previousOutput' and 'output' may refer to the same buffer if tiling is not

used, i.e. 'previousOutput' is first read by this function and later overwritten with the denoised result. 'output' can be passed as 'previousOutput' to the next frame. In addition, 'previousOutputInternalGuideLayer' and 'outputInternalGuideLayer' must both be allocated regardless of tiling mode. The pixel format must be OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER and the dimension must be identical to to the other input layers. In the first frame memory in 'previousOutputInternalGuideLayer' must either contain valid data from previous denoiser runs or set to zero. In other model kinds (not temporal) 'previousOutput' and the internal guide layers are ignored.

If OPTIX_DENOISER_MODEL_KIND_TEMPORAL or OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV is selected, the normal vector guide image must be given as 3d vectors in camera space. In the other models only the x and y channels are used and other channels are ignored.

Parameters

in	*denoiser*
in	*stream*
in	*params*
in	*denoiserState*
in	*denoiserStateSizeInBytes*
in	*guideLayer*
in	*layers*
in	*numLayers*
in	*inputOffsetX*
in	*inputOffsetY*
in	*scratch*
in	*scratchSizeInBytes*

8.17.2.17 optixDenoiserSetup()

```
OptixResult optixDenoiserSetup (
            OptixDenoiser denoiser,
            CUstream stream,
            unsigned int inputWidth,
            unsigned int inputHeight,
            CUdeviceptr denoiserState,
            size_t denoiserStateSizeInBytes,
            CUdeviceptr scratch,
            size_t scratchSizeInBytes )
```

Initializes the state required by the denoiser.

'inputWidth' and 'inputHeight' must include overlap on both sides of the image if tiling is being used. The overlap is returned by optixDenoiserComputeMemoryResources. For subsequent calls to optixDenoiserInvoke 'inputWidth' and 'inputHeight' are the maximum dimensions of the input layers. Dimensions of the input layers passed to optixDenoiserInvoke may be different in each invocation however they always must be smaller than 'inputWidth' and 'inputHeight' passed to optixDenoiserSetup.

Parameters

in	*denoiser*
in	*stream*
in	*inputWidth*
in	*inputHeight*
in	*denoiserState*
in	*denoiserStateSizeInBytes*
in	*scratch*
in	*scratchSizeInBytes*

8.17.2.18 optixDeviceContextCreate()

```
OptixResult optixDeviceContextCreate (
            CUcontext fromContext,
            const OptixDeviceContextOptions * options,
            OptixDeviceContext * context )
```

Create a device context associated with the CUDA context specified with 'fromContext'.

If zero is specified for 'fromContext', OptiX will use the current CUDA context. The CUDA context should be initialized before calling optixDeviceContextCreate.

Parameters

in	*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

8.17.2.19 optixDeviceContextDestroy()

```
OptixResult optixDeviceContextDestroy (
            OptixDeviceContext context )
```

Destroys all CPU and GPU state associated with the device.

It will attempt to block on CUDA streams that have launch work outstanding.

Any API objects, such as OptixModule and OptixPipeline, not already destroyed will be destroyed.

Thread safety: A device context must not be destroyed while it is still in use by concurrent API calls in other threads.

8.17.2.20 optixDeviceContextGetCacheDatabaseSizes()

```
OptixResult optixDeviceContextGetCacheDatabaseSizes (
            OptixDeviceContext context,
            size_t * lowWaterMark,
            size_t * highWaterMark )
```

Returns the low and high water marks for disk cache garbage collection. If the cache has been disabled by setting the environment variable OPTIX_CACHE_MAXSIZE=0, this function will return 0 for the low and high water marks.

Parameters

in	*context*	the device context
out	*lowWaterMark*	the low water mark
out	*highWaterMark*	the high water mark

8.17.2.21 optixDeviceContextGetCacheEnabled()

```
OptixResult optixDeviceContextGetCacheEnabled (
            OptixDeviceContext context,
            int * enabled )
```

Indicates whether the disk cache is enabled or disabled.

Parameters

in	*context*	the device context
out	*enabled*	1 if enabled, 0 if disabled

8.17.2.22 optixDeviceContextGetCacheLocation()

```
OptixResult optixDeviceContextGetCacheLocation (
            OptixDeviceContext context,
            char * location,
            size_t locationSize )
```

Returns the location of the disk cache. If the cache has been disabled by setting the environment variable OPTIX_CACHE_MAXSIZE=0, this function will return an empy string.

Parameters

in	*context*	the device context
out	*location*	directory of disk cache, null terminated if locationSize > 0
in	*locationSize*	locationSize

8.17.2.23 optixDeviceContextGetProperty()

```
OptixResult optixDeviceContextGetProperty (
            OptixDeviceContext context,
```

```
          OptixDeviceProperty property,
          void * value,
          size_t sizeInBytes )
```

Query properties of a device context.

Parameters

in	*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

8.17.2.24 optixDeviceContextSetCacheDatabaseSizes()

```
OptixResult optixDeviceContextSetCacheDatabaseSizes (
          OptixDeviceContext context,
          size_t lowWaterMark,
          size_t highWaterMark )
```

Sets the low and high water marks for disk cache garbage collection.

Garbage collection is triggered when a new entry is written to the cache and the current cache data size plus the size of the cache entry that is about to be inserted exceeds the high water mark. Garbage collection proceeds until the size reaches the low water mark. Garbage collection will always free enough space to insert the new entry without exceeding the low water mark. Setting either limit to zero will disable garbage collection. An error will be returned if both limits are non-zero and the high water mark is smaller than the low water mark.

Note that garbage collection is performed only on writes to the disk cache. No garbage collection is triggered on disk cache initialization or immediately when calling this function, but on subsequent inserting of data into the database.

If the size of a compiled module exceeds the value configured for the high water mark and garbage collection is enabled, the module will not be added to the cache and a warning will be added to the log.

The high water mark can be overridden with the environment variable OPTIX_CACHE_MAXSIZE. The environment variable takes precedence over the function parameters. The low water mark will be set to half the value of OPTIX_CACHE_MAXSIZE. Setting OPTIX_CACHE_MAXSIZE to 0 will disable the disk cache, but will not alter the contents of the cache. Negative and non-integer values will be ignored.

Parameters

in	*context*	the device context
in	*lowWaterMark*	the low water mark
in	*highWaterMark*	the high water mark

8.17.2.25 optixDeviceContextSetCacheEnabled()

```
OptixResult optixDeviceContextSetCacheEnabled (
          OptixDeviceContext context,
```

```
            int enabled )
```

Enables or disables the disk cache.

If caching was previously disabled, enabling it will attempt to initialize the disk cache database using the currently configured cache location. An error will be returned if initialization fails.

Note that no in-memory cache is used, so no caching behavior will be observed if the disk cache is disabled.

The cache can be disabled by setting the environment variable OPTIX_CACHE_MAXSIZE=0. The environment variable takes precedence over this setting. See optixDeviceContextSetCacheDatabaseSizes for additional information.

Note that the disk cache can be disabled by the environment variable, but it cannot be enabled via the environment if it is disabled via the API.

Parameters

in	*context*	the device context
in	*enabled*	1 to enabled, 0 to disable

8.17.2.26 optixDeviceContextSetCacheLocation()

```
OptixResult optixDeviceContextSetCacheLocation (
            OptixDeviceContext context,
            const char * location )
```

Sets the location of the disk cache.

The location is specified by a directory. This directory should not be used for other purposes and will be created if it does not exist. An error will be returned if 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

8.17.2.27 optixDeviceContextSetLogCallback()

```
OptixResult optixDeviceContextSetLogCallback (
```

```
OptixDeviceContext context,
OptixLogCallback callbackFunction,
void * callbackData,
unsigned int callbackLevel )
```

Sets the current log callback method.

See OptixLogCallback for more details.

Thread safety: It is guaranteed that the callback itself (callbackFunction and callbackData) are updated atomically. It is not guaranteed that the callback itself (callbackFunction and callbackData) and the callbackLevel are updated atomically. It is unspecified when concurrent API calls using the same context start to make use of the new callback method.

Parameters

in	*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

8.17.2.28 optixDisplacementMicromapArrayBuild()

```
OptixResult optixDisplacementMicromapArrayBuild (
OptixDeviceContext context,
CUstream stream,
const OptixDisplacementMicromapArrayBuildInput * buildInput,
const OptixMicromapBuffers * buffers )
```

FIXME Construct an array of Displacement Micromap (DMMs).

Each triangle within a DMM GAS geometry references one DMM that specifies how to subdivide it into micro-triangles. A DMM gives a subdivision resolution into 4^N micro-triangles, and displacement values for each of the vertices in the subdivided mesh. The values are combined with e.g. normal vectors, scale and bias given as AS build inputs, to get the final geometry. A DMM is encoded in one or more compressed blocks, each block having displacement values for a subtriangle of 64..1024 micro-triangles.

Parameters

in	*context*	
in	*stream*	
in	*buildInput*	a single build input object referencing many DMMs
in	*buffers*	the buffers used for build

8.17.2.29 optixDisplacementMicromapArrayComputeMemoryUsage()

```
OptixResult optixDisplacementMicromapArrayComputeMemoryUsage (
OptixDeviceContext context,
const OptixDisplacementMicromapArrayBuildInput * buildInput,
OptixMicromapBufferSizes * bufferSizes )
```

Determine the amount of memory necessary for a Displacement Micromap Array build.

Parameters

in	*context*
in	*buildInput*
out	*bufferSizes*

8.17.2.30 optixGetErrorName()

```
const char * optixGetErrorName (
          OptixResult result )
```

Returns a string containing the name of an error code in the enum.

Output is a string representation of the enum. For example "OPTIX_SUCCESS" for OPTIX_SUCCESS and "OPTIX_ERROR_INVALID_VALUE" for OPTIX_ERROR_INVALID_VALUE.

If the error code is not recognized, "Unrecognized OptixResult code" is returned.

Parameters

in	*result*	OptixResult enum to generate string name for

See also optixGetErrorString

8.17.2.31 optixGetErrorString()

```
const char * optixGetErrorString (
          OptixResult result )
```

Returns the description string for an error code.

Output is a string description of the enum. For example "Success" for OPTIX_SUCCESS and "Invalid value" for OPTIX_ERROR_INVALID_VALUE.

If the error code is not recognized, "Unrecognized OptixResult code" is returned.

Parameters

in	*result*	OptixResult enum to generate string description for

See also optixGetErrorName

8.17.2.32 optixLaunch()

```
OptixResult optixLaunch (
          OptixPipeline pipeline,
          CUstream stream,
          CUdeviceptr pipelineParams,
          size_t pipelineParamsSize,
          const OptixShaderBindingTable * sbt,
          unsigned int width,
```

```
            unsigned int height,
            unsigned int depth )
```

Where the magic happens.

The stream and pipeline must belong to the same device context. Multiple launches may be issues in parallel from multiple threads to different streams.

pipelineParamsSize number of bytes are copied from the device memory pointed to by pipelineParams before launch. It is an error if pipelineParamsSize is greater than the size of the variable declared in modules and identified by OptixPipelineCompileOptions::pipelineLaunchParamsVariableName. If the launch params variable was optimized out or not found in the modules linked to the pipeline then the pipelineParams and pipelineParamsSize parameters are ignored.

sbt points to the shader binding table, which defines shader groupings and their resources. See the SBT spec.

Parameters

in	*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.

8.17.2.33 optixModuleCreate()

```
OptixResult optixModuleCreate (
            OptixDeviceContext context,
            const OptixModuleCompileOptions * moduleCompileOptions,
            const OptixPipelineCompileOptions * pipelineCompileOptions,
            const char * input,
            size_t inputSize,
            char * logString,
            size_t * logStringSize,
            OptixModule * module )
```

Compiling programs into a module. These programs can be passed in as either PTX or OptiX-IR.

See the Programming Guide for details, as well as how to generate these encodings from CUDA sources.

logString is an optional buffer that contains compiler feedback and errors. This information is also passed to the context logger (if enabled), however it may be difficult to correlate output to the logger to specific API invocations when using multiple threads. The output to logString will only contain feedback for this specific invocation of this API call.

logStringSize as input should be a pointer to the number of bytes backing logString. Upon return it contains the length of the log message (including the null terminator) which may be greater than the input value. In this case, the log message will be truncated to fit into logString.

If logString or logStringSize are NULL, no output is written to logString. If logStringSize points to a value that is zero, no output is written. This does not affect output to the context logger if enabled.

Parameters

in	*context*	
in	*moduleCompileOptions*	
in	*pipelineCompileOptions*	All modules in a pipeline need to use the same values for the pipeline compile options.
in	*input*	Pointer to the input code.
in	*inputSize*	Parsing proceeds up to inputSize characters. Or, when reading PTX input, 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, input is 0, module is 0.

8.17.2.34 optixModuleCreateWithTasks()

```
OptixResult optixModuleCreateWithTasks (
            OptixDeviceContext context,
            const OptixModuleCompileOptions * moduleCompileOptions,
            const OptixPipelineCompileOptions * pipelineCompileOptions,
            const char * input,
            size_t inputSize,
            char * logString,
            size_t * logStringSize,
            OptixModule * module,
            OptixTask * firstTask )
```

This function is designed to do just enough work to create the OptixTask return parameter and is expected to be fast enough run without needing parallel execution. A single thread could generate all the OptixTask objects for further processing in a work pool.

Options are similar to optixModuleCreate( ), aside from the return parameter, firstTask.

The memory used to hold the input should be live until all tasks are finished.

It is illegal to call optixModuleDestroy( ) if any OptixTask objects are currently being executed. In that case OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE will be returned.

If an invocation of optixTaskExecute fails, the OptixModule will be marked as OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE if there are outstanding tasks or OPTIX_MODULE_

COMPILE_STATE_FAILURE if there are no outstanding tasks. Subsequent calls to optixTaskExecute( ) may execute additional work to collect compilation errors generated from the input. Currently executing tasks will not necessarily be terminated immediately but at the next opportunity. Logging will continue to be directed to the logger installed with the OptixDeviceContext. If logString is provided to optixModuleCreateWithTasks( ), it will contain all the compiler feedback from all executed tasks. The lifetime of the memory pointed to by logString should extend from calling optixModuleCreateWithTasks( ) to when the compilation state is either OPTIX_MODULE_COMPILE_STATE_FAILURE or OPTIX_MODULE_COMPILE_STATE_COMPLETED. OptiX will not write to the logString outside of execution of optixModuleCreateWithTasks( ) or optixTaskExecute( ). If the compilation state is OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE and no further execution of optixTaskExecute( ) is performed the logString may be reclaimed by the application before calling optixModuleDestroy( ). The contents of logString will contain output from currently completed tasks. All OptixTask objects associated with a given OptixModule will be cleaned up when optixModuleDestroy( ) is called regardless of whether the compilation was successful or not. If the compilation state is OPTIX_MODULE_COMPILE_STATE_IMPENDIND_FAILURE, any unstarted OptixTask objects do not need to be executed though there is no harm doing so.

See also optixModuleCreate

8.17.2.35 optixModuleDestroy( )

```
OptixResult optixModuleDestroy (
          OptixModule module )
```

Call for OptixModule objects created with optixModuleCreate and optixModuleDeserialize.

Modules must not be destroyed while they are still used by any program group.

Thread safety: A module must not be destroyed while it is still in use by concurrent API calls in other threads.

8.17.2.36 optixModuleGetCompilationState( )

```
OptixResult optixModuleGetCompilationState (
          OptixModule module,
          OptixModuleCompileState * state )
```

When creating a module with tasks, the current state of the module can be queried using this function.

Thread safety: Safe to call from any thread until optixModuleDestroy is called.

See also optixModuleCreateWithTasks

8.17.2.37 optixOpacityMicromapArrayBuild( )

```
OptixResult optixOpacityMicromapArrayBuild (
          OptixDeviceContext context,
          CUstream stream,
          const OptixOpacityMicromapArrayBuildInput * buildInput,
          const OptixMicromapBuffers * buffers )
```

Construct an array of Opacity Micromaps.

Each triangle within an instance/GAS may reference one opacity micromap to give finer control over alpha behavior. A opacity micromap consists of a set of 4^N micro-triangles in a triangular uniform barycentric grid. Multiple opacity micromaps are collected (built) into a opacity micromap array with this function. Each geometry in a GAS may bind a single opacity micromap array and can use opacity

micromaps from that array only.

Each micro-triangle within a opacity micromap can be in one of four states: Transparent, Opaque, Unknown-Transparent or Unknown-Opaque. During traversal, if a triangle with a opacity micromap attached is intersected, the opacity micromap is queried to categorize the hit as either opaque, unknown (alpha) or a miss. Geometry, ray or instance flags that modify the alpha/opaque behavior are applied *after* this opacity micromap query.

The opacity micromap query may operate in 2-state mode (alpha testing) or 4-state mode (AHS culling), depending on the opacity micromap type and ray/instance flags. When operating in 2-state mode, alpha hits will not be reported, and transparent and opaque hits must be accurate.

Parameters

in	*context*	
in	*stream*	
in	*buildInput*	a single build input object referencing many opacity micromaps
in	*buffers*	the buffers used for build

8.17.2.38 optixOpacityMicromapArrayComputeMemoryUsage()

```
OptixResult optixOpacityMicromapArrayComputeMemoryUsage (
            OptixDeviceContext context,
            const OptixOpacityMicromapArrayBuildInput * buildInput,
            OptixMicromapBufferSizes * bufferSizes )
```

Determine the amount of memory necessary for a Opacity Micromap Array build.

Parameters

in	*context*
in	*buildInput*
out	*bufferSizes*

8.17.2.39 optixOpacityMicromapArrayGetRelocationInfo()

```
OptixResult optixOpacityMicromapArrayGetRelocationInfo (
            OptixDeviceContext context,
            CUdeviceptr opacityMicromapArray,
            OptixRelocationInfo * info )
```

Obtain relocation information, stored in OptixRelocationInfo, for a given context and opacity micromap array.

The relocation information can be passed to optixCheckRelocationCompatibility to determine if a opacity micromap array, referenced by buffers, can be relocated to a different device's memory space (see optixCheckRelocationCompatibility).

When used with optixOpacityMicromapArrayRelocate, it provides data necessary for doing the relocation.

If the opacity micromap array data associated with 'opacityMicromapArray' is copied multiple times, the same OptixRelocationInfo can also be used on all copies.

Parameters

in	*context*
in	*opacityMicromapArray*
out	*info*

8.17.2.40 optixOpacityMicromapArrayRelocate()

```
OptixResult optixOpacityMicromapArrayRelocate (
            OptixDeviceContext context,
            CUstream stream,
            const OptixRelocationInfo * info,
            CUdeviceptr targetOpacityMicromapArray,
            size_t targetOpacityMicromapArraySizeInBytes )
```

optixOpacityMicromapArrayRelocate is called to update the opacity micromap array after it has been relocated. Relocation is necessary when the opacity micromap array's location in device memory has changed. optixOpacityMicromapArrayRelocate does not copy the memory. This function only operates on the relocated memory whose new location is specified by 'targetOpacityMicromapArray'. The original memory (source) is not required to be valid, only the OptixRelocationInfo.

Before calling optixOpacityMicromapArrayRelocate, optixCheckRelocationCompatibility should be called to ensure the copy will be compatible with the destination device context.

The memory pointed to by 'targetOpacityMicromapArray' should be allocated with the same size as the source opacity micromap array. Similar to the 'OptixMicromapBuffers::output' used in optixOpacityMicromapArrayBuild, this pointer must be a multiple of OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT.

The memory in 'targetOpacityMicromapArray' must be allocated as long as the opacity micromap array is in use.

Note that any Acceleration Structures build using the original memory (source) as input will still be associated with this original memory. To associate an existing (possibly relocated) Acceleration Structures with the relocated opacity micromap array, use optixAccelBuild to update the existing Acceleration Structures (See OPTIX_BUILD_OPERATION_UPDATE)

Parameters

in	*context*
in	*stream*
in	*info*
in	*targetOpacityMicromapArray*
in	*targetOpacityMicromapArraySizeInBytes*

8.17.2.41 optixPipelineCreate()

```
OptixResult optixPipelineCreate (
            OptixDeviceContext context,
            const OptixPipelineCompileOptions * pipelineCompileOptions,
            const OptixPipelineLinkOptions * pipelineLinkOptions,
```

```
const OptixProgramGroup * programGroups,
unsigned int numProgramGroups,
char * logString,
size_t * logStringSize,
OptixPipeline * pipeline )
```

logString is an optional buffer that contains compiler feedback and errors. This information is also passed to the context logger (if enabled), however it may be difficult to correlate output to the logger to specific API invocations when using multiple threads. The output to logString will only contain feedback for this specific invocation of this API call.

logStringSize as input should be a pointer to the number of bytes backing logString. Upon return it contains the length of the log message (including the null terminator) which may be greater than the input value. In this case, the log message will be truncated to fit into logString.

If logString or logStringSize are NULL, no output is written to logString. If logStringSize points to a value that is zero, no output is written. This does not affect output to the context logger if enabled.

Parameters

in	*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*	
out	*pipeline*	

8.17.2.42 optixPipelineDestroy()

```
OptixResult optixPipelineDestroy (
        OptixPipeline pipeline )
```

Thread safety: A pipeline must not be destroyed while it is still in use by concurrent API calls in other threads.

8.17.2.43 optixPipelineSetStackSize()

```
OptixResult optixPipelineSetStackSize (
        OptixPipeline pipeline,
        unsigned int directCallableStackSizeFromTraversal,
        unsigned int directCallableStackSizeFromState,
        unsigned int continuationStackSize,
        unsigned int maxTraversableGraphDepth )
```

Sets the stack sizes for a pipeline.

Users are encouraged to see the programming guide and the implementations of the helper functions to understand how to construct the stack sizes based on their particular needs.

If this method is not used, an internal default implementation is used. The default implementation is correct (but not necessarily optimal) as long as the maximum depth of call trees of CC programs is at most 2, and no DC programs or motion transforms are used.

The maxTraversableGraphDepth responds to the maximal number of traversables visited when calling trace. Every acceleration structure and motion transform count as one level of traversal. E.g., for a simple IAS (instance acceleration structure) -> GAS (geometry acceleration structure) traversal graph, the maxTraversableGraphDepth is two. For IAS -> MT (motion transform) -> GAS, the maxTraversableGraphDepth is three. Note that it does not matter whether a IAS or GAS has motion or not, it always counts as one. Launching optix with exceptions turned on (see OPTIX_EXCEPTION_FLAG_TRACE_DEPTH) will throw an exception if the specified maxTraversableGraphDepth is too small.

Parameters

in	*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.

8.17.2.44 optixProgramGroupCreate()

```
OptixResult optixProgramGroupCreate (
            OptixDeviceContext context,
            const OptixProgramGroupDesc * programDescriptions,
            unsigned int numProgramGroups,
            const OptixProgramGroupOptions * options,
            char * logString,
            size_t * logStringSize,
            OptixProgramGroup * programGroups )
```

logString is an optional buffer that contains compiler feedback and errors. This information is also passed to the context logger (if enabled), however it may be difficult to correlate output to the logger to specific API invocations when using multiple threads. The output to logString will only contain feedback for this specific invocation of this API call.

logStringSize as input should be a pointer to the number of bytes backing logString. Upon return it contains the length of the log message (including the null terminator) which may be greater than the input value. In this case, the log message will be truncated to fit into logString.

If logString or logStringSize are NULL, no output is written to logString. If logStringSize points to a value that is zero, no output is written. This does not affect output to the context logger if enabled.

Creates numProgramGroups OptiXProgramGroup objects from the specified OptixProgramGroupDesc array. The size of the arrays must match.

Parameters

in	*context*	

Parameters

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*	

8.17.2.45 optixProgramGroupDestroy()

```
OptixResult optixProgramGroupDestroy (
            OptixProgramGroup programGroup )
```

Thread safety: A program group must not be destroyed while it is still in use by concurrent API calls in other threads.

8.17.2.46 optixProgramGroupGetStackSize()

```
OptixResult optixProgramGroupGetStackSize (
            OptixProgramGroup programGroup,
            OptixStackSizes * stackSizes,
            OptixPipeline pipeline )
```

Returns the stack sizes for the given program group. When programs in this `programGroup` are relying on external functions, the corresponding stack sizes can only be correctly retrieved when all functions are known after linking, i.e. when a pipeline has been created. When `pipeline` is set to NULL, the stack size will be calculated excluding external functions. In this case a warning will be issued if external functions are referenced by the OptixModule.

Parameters

in	*programGroup*	the program group
out	*stackSizes*	the corresponding stack sizes
in	*pipeline*	considering the program group within the given pipeline, can be NULL

8.17.2.47 optixSbtRecordPackHeader()

```
OptixResult optixSbtRecordPackHeader (
            OptixProgramGroup programGroup,
            void * sbtRecordHeaderHostPointer )
```

Parameters

in	*programGroup*	the program group containing the program(s)
out	*sbtRecordHeaderHostPointer*	the result sbt record header

8.17.2.48 optixTaskExecute()

```
OptixResult optixTaskExecute (
            OptixTask task,
            OptixTask * additionalTasks,
            unsigned int maxNumAdditionalTasks,
            unsigned int * numAdditionalTasksCreated )
```

Each OptixTask should be executed with optixTaskExecute( ). If additional parallel work is found, new OptixTask objects will be returned in additionalTasks along with the number of additional tasks in numAdditionalTasksCreated. The parameter additionalTasks should point to a user allocated array of minimum size maxNumAdditionalTasks. OptiX can generate upto maxNumAdditionalTasks additional tasks.

Each task can be executed in parallel and in any order.

Thread safety: Safe to call from any thread until optixModuleDestroy( ) is called for any associated task.

See also optixModuleCreateWithTasks

Parameters

in	*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

8.18 optix_host.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2023 NVIDIA Corporation.  All rights reserved.
3 *
4 * NVIDIA Corporation and its licensors retain all intellectual property and proprietary
5 * rights in and to this software, related documentation and any modifications thereto.
6 * Any use, reproduction, disclosure or distribution of this software and related
7 * documentation without an express license agreement from NVIDIA Corporation is strictly
8 * prohibited.
9 *
10 * TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THIS SOFTWARE IS PROVIDED *AS IS*
11 * AND NVIDIA AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED,
12 * INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
13 * PARTICULAR PURPOSE.  IN NO EVENT SHALL NVIDIA OR ITS SUPPLIERS BE LIABLE FOR ANY
14 * SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT
15 * LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF
16 * BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR
17 * INABILITY TO USE THIS SOFTWARE, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF
18 * SUCH DAMAGES
19 */
20
27
28 #ifndef OPTIX_OPTIX_HOST_H
29 #define OPTIX_OPTIX_HOST_H
30
31 #include "optix_types.h"
32 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
33 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver types must be defined through other
34 // means before including optix headers.
35 #include <cuda.h>
```

```
36 #endif
37
38 #ifdef NV_MODULE_OPTIX
39 // This is a mechanism to include <g_nvconfig.h> in driver builds only and translate any nvconfig macro to
a custom OPTIX-specific macro, that can also be used in SDK builds/installs
40 #include <exp/misc/optix_nvconfig_translate.h>  // includes <g_nvconfig.h>
41 #endif  // NV_MODULE_OPTIX
42
43
44 #ifdef __cplusplus
45 extern "C" {
46 #endif
47
50
53
54
65 const char* optixGetErrorName(OptixResult result);
66
77 const char* optixGetErrorString(OptixResult result);
78
80
82
83
102 OptixResult optixDeviceContextCreate(CUcontext fromContext, const OptixDeviceContextOptions* options,
OptixDeviceContext* context);
103
112 OptixResult optixDeviceContextDestroy(OptixDeviceContext context);
113
120 OptixResult optixDeviceContextGetProperty(OptixDeviceContext context, OptixDeviceProperty property,
void* value, size_t sizeInBytes);
121
135 OptixResult optixDeviceContextSetLogCallback(OptixDeviceContext context,
136                                              OptixLogCallback   callbackFunction,
137                                              void*              callbackData,
138                                              unsigned int       callbackLevel);
139
158 OptixResult optixDeviceContextSetCacheEnabled(OptixDeviceContext context,
159                                               int                enabled);
160
181 OptixResult optixDeviceContextSetCacheLocation(OptixDeviceContext context, const char* location);
182
210 OptixResult optixDeviceContextSetCacheDatabaseSizes(OptixDeviceContext context, size_t lowWaterMark,
size_t highWaterMark);
211
216 OptixResult optixDeviceContextGetCacheEnabled(OptixDeviceContext context, int* enabled);
223 OptixResult optixDeviceContextGetCacheLocation(OptixDeviceContext context, char* location, size_t
locationSize);
224
232 OptixResult optixDeviceContextGetCacheDatabaseSizes(OptixDeviceContext context, size_t* lowWaterMark,
size_t* highWaterMark);
233
235
237
238
262 OptixResult optixPipelineCreate(OptixDeviceContext                 context,
263                                 const OptixPipelineCompileOptions* pipelineCompileOptions,
264                                 const OptixPipelineLinkOptions*    pipelineLinkOptions,
265                                 const OptixProgramGroup*           programGroups,
266                                 unsigned int                       numProgramGroups,
267                                 char*                              logString,
268                                 size_t*                            logStringSize,
269                                 OptixPipeline*                     pipeline);
270
272 OptixResult optixPipelineDestroy(OptixPipeline pipeline);
273
296 OptixResult optixPipelineSetStackSize(OptixPipeline pipeline,
297                                       unsigned int  directCallableStackSizeFromTraversal,
```

```
298                                     unsigned int  directCallableStackSizeFromState,
299                                     unsigned int  continuationStackSize,
300                                     unsigned int  maxTraversableGraphDepth);
301
303
305
306
336 OptixResult optixModuleCreate(OptixDeviceContext                 context,
337                                 const OptixModuleCompileOptions*   moduleCompileOptions,
338                                 const OptixPipelineCompileOptions* pipelineCompileOptions,
339                                 const char*                        input,
340                                 size_t                             inputSize,
341                                 char*                              logString,
342                                 size_t*                            logStringSize,
343                                 OptixModule*                       module);
344
364
377
385 OptixResult optixModuleCreateWithTasks(OptixDeviceContext                 context,
386                                        const OptixModuleCompileOptions*   moduleCompileOptions,
387                                        const OptixPipelineCompileOptions* pipelineCompileOptions,
388                                        const char*                        input,
389                                        size_t                             inputSize,
390                                        char*                              logString,
391                                        size_t*                            logStringSize,
392                                        OptixModule*                       module,
393                                        OptixTask*                         firstTask);
394
401 OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState* state);
402
408 OptixResult optixModuleDestroy(OptixModule module);
409
413 OptixResult optixBuiltinISModuleGet(OptixDeviceContext                 context,
414                                     const OptixModuleCompileOptions*   moduleCompileOptions,
415                                     const OptixPipelineCompileOptions* pipelineCompileOptions,
416                                     const OptixBuiltinISOptions*       builtinISOptions,
417                                     OptixModule*                       builtinModule);
418
420
422
423
441 OptixResult optixTaskExecute(OptixTask task, OptixTask* additionalTasks, unsigned int
maxNumAdditionalTasks, unsigned int* numAdditionalTasksCreated);
442
444
446
447
456 OptixResult optixProgramGroupGetStackSize(OptixProgramGroup programGroup, OptixStackSizes* stackSizes,
OptixPipeline pipeline);
457
483 OptixResult optixProgramGroupCreate(OptixDeviceContext             context,
484                                     const OptixProgramGroupDesc*    programDescriptions,
485                                     unsigned int                    numProgramGroups,
486                                     const OptixProgramGroupOptions* options,
487                                     char*                           logString,
488                                     size_t*                         logStringSize,
489                                     OptixProgramGroup*              programGroups);
490
492 OptixResult optixProgramGroupDestroy(OptixProgramGroup programGroup);
493
495
497
498
525 OptixResult optixLaunch(OptixPipeline                pipeline,
526                         CUstream                     stream,
527                         CUdeviceptr                  pipelineParams,
528                         size_t                       pipelineParamsSize,
```

```
529                         const OptixShaderBindingTable* sbt,
530                         unsigned int                   width,
531                         unsigned int                   height,
532                         unsigned int                   depth);
533
536 OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void* sbtRecordHeaderHostPointer);
537
539
541
542
548 OptixResult optixAccelComputeMemoryUsage(OptixDeviceContext           context,
549                                          const OptixAccelBuildOptions* accelOptions,
550                                          const OptixBuildInput*        buildInputs,
551                                          unsigned int                  numBuildInputs,
552                                          OptixAccelBufferSizes*        bufferSizes);
553
566 OptixResult optixAccelBuild(OptixDeviceContext           context,
567                             CUstream                     stream,
568                             const OptixAccelBuildOptions* accelOptions,
569                             const OptixBuildInput*        buildInputs,
570                             unsigned int                  numBuildInputs,
571                             CUdeviceptr                   tempBuffer,
572                             size_t                        tempBufferSizeInBytes,
573                             CUdeviceptr                   outputBuffer,
574                             size_t                        outputBufferSizeInBytes,
575                             OptixTraversableHandle*       outputHandle,
576                             const OptixAccelEmitDesc*     emittedProperties,
577                             unsigned int                  numEmittedProperties);
578
596 OptixResult optixAccelGetRelocationInfo(OptixDeviceContext context, OptixTraversableHandle handle,
OptixRelocationInfo* info);
597
609 OptixResult optixCheckRelocationCompatibility(OptixDeviceContext context, const OptixRelocationInfo*
info, int* compatible);
610
648 OptixResult optixAccelRelocate(OptixDeviceContext         context,
649                                CUstream                   stream,
650                                const OptixRelocationInfo* info,
651                                const OptixRelocateInput*  relocateInputs,
652                                size_t                     numRelocateInputs,
653                                CUdeviceptr                targetAccel,
654                                size_t                     targetAccelSizeInBytes,
655                                OptixTraversableHandle*    targetHandle);
656
674 OptixResult optixAccelCompact(OptixDeviceContext      context,
675                               CUstream                stream,
676                               OptixTraversableHandle  inputHandle,
677                               CUdeviceptr             outputBuffer,
678                               size_t                  outputBufferSizeInBytes,
679                               OptixTraversableHandle* outputHandle);
680
689 OptixResult optixAccelEmitProperty(OptixDeviceContext        context,
690                                    CUstream                  stream,
691                                    OptixTraversableHandle    handle,
692                                    const OptixAccelEmitDesc* emittedProperty);
693
698 OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext      onDevice,
699                                                    CUdeviceptr             pointer,
700                                                    OptixTraversableType    traversableType,
701                                                    OptixTraversableHandle* traversableHandle);
702
703
709 OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext                     context,
710                                                         const OptixOpacityMicromapArrayBuildInput* buildInput,
711                                                         OptixMicromapBufferSizes*                bufferSizes);
712
735 OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext                       context,
```

```
736                                             CUstream                                    stream,
737                                             const OptixOpacityMicromapArrayBuildInput* buildInput,
738                                             const OptixMicromapBuffers*                 buffers);
739
755 OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext context, CUdeviceptr
opacityMicromapArray, OptixRelocationInfo* info);
756
783 OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext         context,
784                                               CUstream                   stream,
785                                               const OptixRelocationInfo* info,
786                                               CUdeviceptr                targetOpacityMicromapArray,
787                                               size_t
targetOpacityMicromapArraySizeInBytes);
788
794 OptixResult optixDisplacementMicromapArrayComputeMemoryUsage(OptixDeviceContext
context,
795                                                              const
OptixDisplacementMicromapArrayBuildInput* buildInput,
796                                                              OptixMicromapBufferSizes* bufferSizes);
797
811 OptixResult optixDisplacementMicromapArrayBuild(OptixDeviceContext                              context,
812                                                 CUstream                                        stream,
813                                                 const OptixDisplacementMicromapArrayBuildInput* buildInput,
814                                                 const OptixMicromapBuffers*                     buffers);
815
816
818
820
821
833 OptixResult optixDenoiserCreate(OptixDeviceContext context,
834                                 OptixDenoiserModelKind modelKind,
835                                 const OptixDenoiserOptions* options,
836                                 OptixDenoiser* denoiser);
837
850 OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context,
851                                              const void* userData, size_t userDataSizeInBytes,
OptixDenoiser* denoiser);
852
854 OptixResult optixDenoiserDestroy(OptixDenoiser denoiser);
855
875 OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser denoiser,
876                                                 unsigned int        outputWidth,
877                                                 unsigned int        outputHeight,
878                                                 OptixDenoiserSizes* returnSizes);
879
896 OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
897                                CUstream      stream,
898                                unsigned int  inputWidth,
899                                unsigned int  inputHeight,
900                                CUdeviceptr   denoiserState,
901                                size_t        denoiserStateSizeInBytes,
902                                CUdeviceptr   scratch,
903                                size_t        scratchSizeInBytes);
904
970 OptixResult optixDenoiserInvoke(OptixDenoiser                   denoiser,
971                                 CUstream                        stream,
972                                 const OptixDenoiserParams*      params,
973                                 CUdeviceptr                     denoiserState,
974                                 size_t                          denoiserStateSizeInBytes,
975                                 const OptixDenoiserGuideLayer*  guideLayer,
976                                 const OptixDenoiserLayer*       layers,
977                                 unsigned int                    numLayers,
978                                 unsigned int                    inputOffsetX,
979                                 unsigned int                    inputOffsetY,
980                                 CUdeviceptr                     scratch,
981                                 size_t                          scratchSizeInBytes);
982
```

```
1006 OptixResult optixDenoiserComputeIntensity(OptixDenoiser       denoiser,
1007                                           CUstream            stream,
1008                                           const OptixImage2D* inputImage,
1009                                           CUdeviceptr         outputIntensity,
1010                                           CUdeviceptr         scratch,
1011                                           size_t              scratchSizeInBytes);
1012
1027 OptixResult optixDenoiserComputeAverageColor(OptixDenoiser       denoiser,
1028                                              CUstream            stream,
1029                                              const OptixImage2D* inputImage,
1030                                              CUdeviceptr         outputAverageColor,
1031                                              CUdeviceptr         scratch,
1032                                              size_t              scratchSizeInBytes);
1033
1035
1036 #ifdef __cplusplus
1037 }
1038 #endif
1039
1040 #include "optix_function_table.h"
1041
1042 #endif  // OPTIX_OPTIX_HOST_H
```

8.19 optix_micromap.h File Reference

Functions

- OPTIX_MICROMAP_INLINE_FUNC void optixMicromapIndexToBaseBarycentrics (unsigned int micromapTriangleIndex, unsigned int subdivisionLevel, float2 &baseBarycentrics0, float2 &baseBarycentrics1, float2 &baseBarycentrics2)
- OPTIX_MICROMAP_INLINE_FUNC float2 optixBaseBarycentricsToMicroBarycentrics (float2 baseBarycentrics, float2 microVertexBaseBarycentrics[3])

8.19.1 Detailed Description

OptiX micromap helper functions.

Author

NVIDIA Corporation

OptiX micromap helper functions. Useable on either host or device.

8.19.2 Function Documentation

8.19.2.1 optixBaseBarycentricsToMicroBarycentrics()

```
OPTIX_MICROMAP_INLINE_FUNC float2 optixBaseBarycentricsToMicroBarycentrics (
          float2 baseBarycentrics,
          float2 microVertexBaseBarycentrics[3] )
```

Maps barycentrics in the space of the base triangle to barycentrics of a micro triangle. The vertices of the micro triangle are defined by its barycentrics in the space of the base triangle. These can be queried for a DMM hit by using optixGetMicroTriangleBarycentricsData().

8.19.2.2 optixMicromapIndexToBaseBarycentrics()

```
OPTIX_MICROMAP_INLINE_FUNC void optixMicromapIndexToBaseBarycentrics (
          unsigned int micromapTriangleIndex,
```

```
unsigned int subdivisionLevel,
float2 & baseBarycentrics0,
float2 & baseBarycentrics1,
float2 & baseBarycentrics2 )
```

Converts a micromap triangle index to the three base-triangle barycentric coordinates of the micro-triangle vertices in the base triangle. The base triangle is the triangle that the micromap is applied to. Note that for displaced micro-meshes this function can be used to compute a UV mapping from sub triangle to base triangle.

Parameters

in	*micromapTriangleIndex*	Index of a micro- or sub triangle within a micromap.
in	*subdivisionLevel*	Number of subdivision levels of the micromap or number of subdivision levels being considered (for sub triangles).
out	*baseBarycentrics0*	Barycentric coordinates in the space of the base triangle of vertex 0 of the micromap triangle.
out	*baseBarycentrics1*	Barycentric coordinates in the space of the base triangle of vertex 1 of the micromap triangle.
out	*baseBarycentrics2*	Barycentric coordinates in the space of the base triangle of vertex 2 of the micromap triangle.

8.20 optix_micromap.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2023 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
37 #ifndef OPTIX_OPTIX_MICROMAP_H
38 #define OPTIX_OPTIX_MICROMAP_H
39
40 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
41 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver type float2 must be defined through other
```

```
42 // means before including optix headers.
43 #include <vector_types.h>
44 #endif
45 #include "internal/optix_micromap_impl.h"
46
56 OPTIX_MICROMAP_INLINE_FUNC void optixMicromapIndexToBaseBarycentrics(unsigned int micromapTriangleIndex,
57                                                                       unsigned int subdivisionLevel,
58                                                                       float2&      baseBarycentrics0,
59                                                                       float2&      baseBarycentrics1,
60                                                                       float2&      baseBarycentrics2)
61 {
62     optix_impl::micro2bary(micromapTriangleIndex, subdivisionLevel, baseBarycentrics0, baseBarycentrics1,
baseBarycentrics2);
63 }
64
68 OPTIX_MICROMAP_INLINE_FUNC float2 optixBaseBarycentricsToMicroBarycentrics(float2 baseBarycentrics,
69                                                                            float2 microVertexBaseBarycentrics[3])
70 {
71     return optix_impl::base2micro(baseBarycentrics, microVertexBaseBarycentrics);
72 }
73
74 #endif  // OPTIX_OPTIX_MICROMAP_H
```

8.21 optix_stack_size.h File Reference

Functions

- OptixResult optixUtilAccumulateStackSizes (OptixProgramGroup programGroup, OptixStackSizes *stackSizes, OptixPipeline pipeline)
- OptixResult optixUtilComputeStackSizes (const OptixStackSizes *stackSizes, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepth, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesDCSplit (const OptixStackSizes *stackSizes, unsigned int dssDCFromTraversal, unsigned int dssDCFromState, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepthFromTraversal, unsigned int maxDCDepthFromState, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesCssCCTree (const OptixStackSizes *stackSizes, unsigned int cssCCTree, unsigned int maxTraceDepth, unsigned int maxDCDepth, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesSimplePathTracer (OptixProgramGroup programGroupRG, OptixProgramGroup programGroupMS1, const OptixProgramGroup *programGroupCH1, unsigned int programGroupCH1Count, OptixProgramGroup programGroupMS2, const OptixProgramGroup *programGroupCH2, unsigned int programGroupCH2Count, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize, OptixPipeline pipeline)

8.21.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.22 optix_stack_size.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2023 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
54 inline OptixResult optixUtilAccumulateStackSizes(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes, OptixPipeline pipeline)
55 {
56     if(!stackSizes)
57         return OPTIX_ERROR_INVALID_VALUE;
58
59     OptixStackSizes localStackSizes;
60     OptixResult     result = optixProgramGroupGetStackSize(programGroup, &localStackSizes, pipeline);
61     if(result != OPTIX_SUCCESS)
62         return result;
63
64     stackSizes->cssRG = std::max(stackSizes->cssRG, localStackSizes.cssRG);
65     stackSizes->cssMS = std::max(stackSizes->cssMS, localStackSizes.cssMS);
66     stackSizes->cssCH = std::max(stackSizes->cssCH, localStackSizes.cssCH);
67     stackSizes->cssAH = std::max(stackSizes->cssAH, localStackSizes.cssAH);
68     stackSizes->cssIS = std::max(stackSizes->cssIS, localStackSizes.cssIS);
69     stackSizes->cssCC = std::max(stackSizes->cssCC, localStackSizes.cssCC);
70     stackSizes->dssDC = std::max(stackSizes->dssDC, localStackSizes.dssDC);
```

```
71
72      return OPTIX_SUCCESS;
73  }
74
88  inline OptixResult optixUtilComputeStackSizes(const OptixStackSizes* stackSizes,
89                                                 unsigned int           maxTraceDepth,
90                                                 unsigned int           maxCCDepth,
91                                                 unsigned int           maxDCDepth,
92                                                unsigned int*          directCallableStackSizeFromTraversal,
93                                                 unsigned int*          directCallableStackSizeFromState,
94                                                 unsigned int*          continuationStackSize)
95  {
96      if(!stackSizes)
97          return OPTIX_ERROR_INVALID_VALUE;
98
99      const unsigned int cssRG = stackSizes->cssRG;
100     const unsigned int cssMS = stackSizes->cssMS;
101     const unsigned int cssCH = stackSizes->cssCH;
102     const unsigned int cssAH = stackSizes->cssAH;
103     const unsigned int cssIS = stackSizes->cssIS;
104     const unsigned int cssCC = stackSizes->cssCC;
105     const unsigned int dssDC = stackSizes->dssDC;
106
107     if(directCallableStackSizeFromTraversal)
108         *directCallableStackSizeFromTraversal = maxDCDepth * dssDC;
109     if(directCallableStackSizeFromState)
110         *directCallableStackSizeFromState = maxDCDepth * dssDC;
111
112     // upper bound on continuation stack used by call trees of continuation callables
113     unsigned int cssCCTree = maxCCDepth * cssCC;
114
115     // upper bound on continuation stack used by CH or MS programs including the call tree of
116     // continuation callables
117     unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
118
119     // clang-format off
120     if(continuationStackSize)
121         *continuationStackSize
122             = cssRG + cssCCTree
123             + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
124             + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
125     // clang-format on
126
127     return OPTIX_SUCCESS;
128 }
129
153 inline OptixResult optixUtilComputeStackSizesDCSplit(const OptixStackSizes* stackSizes,
154                                                      unsigned int           dssDCFromTraversal,
155                                                      unsigned int           dssDCFromState,
156                                                      unsigned int           maxTraceDepth,
157                                                      unsigned int           maxCCDepth,
158                                                      unsigned int           maxDCDepthFromTraversal,
159                                                      unsigned int           maxDCDepthFromState,
160                                                      unsigned int*
directCallableStackSizeFromTraversal,
161                                                  unsigned int*          directCallableStackSizeFromState,
162                                                      unsigned int*          continuationStackSize)
163 {
164     if(!stackSizes)
165         return OPTIX_ERROR_INVALID_VALUE;
166
167     const unsigned int cssRG = stackSizes->cssRG;
168     const unsigned int cssMS = stackSizes->cssMS;
169     const unsigned int cssCH = stackSizes->cssCH;
170     const unsigned int cssAH = stackSizes->cssAH;
171     const unsigned int cssIS = stackSizes->cssIS;
172     const unsigned int cssCC = stackSizes->cssCC;
```

```
173     // use dssDCFromTraversal and dssDCFromState instead of stackSizes->dssDC
174
175     if(directCallableStackSizeFromTraversal)
176         *directCallableStackSizeFromTraversal = maxDCDepthFromTraversal * dssDCFromTraversal;
177     if(directCallableStackSizeFromState)
178         *directCallableStackSizeFromState = maxDCDepthFromState * dssDCFromState;
179
180     // upper bound on continuation stack used by call trees of continuation callables
181     unsigned int cssCCTree = maxCCDepth * cssCC;
182
183     // upper bound on continuation stack used by CH or MS programs including the call tree of
184     // continuation callables
185     unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
186
187     // clang-format off
188     if(continuationStackSize)
189         *continuationStackSize
190             = cssRG + cssCCTree
191             + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
192             + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
193     // clang-format on
194
195     return OPTIX_SUCCESS;
196 }
197
214 inline OptixResult optixUtilComputeStackSizesCssCCTree(const OptixStackSizes* stackSizes,
215                                                        unsigned int           cssCCTree,
216                                                        unsigned int           maxTraceDepth,
217                                                        unsigned int           maxDCDepth,
218                                                        unsigned int*
directCallableStackSizeFromTraversal,
219                                                  unsigned int*         directCallableStackSizeFromState,
220                                                        unsigned int*          continuationStackSize)
221 {
222     if(!stackSizes)
223         return OPTIX_ERROR_INVALID_VALUE;
224
225     const unsigned int cssRG = stackSizes->cssRG;
226     const unsigned int cssMS = stackSizes->cssMS;
227     const unsigned int cssCH = stackSizes->cssCH;
228     const unsigned int cssAH = stackSizes->cssAH;
229     const unsigned int cssIS = stackSizes->cssIS;
230     // use cssCCTree instead of stackSizes->cssCC and maxCCDepth
231     const unsigned int dssDC = stackSizes->dssDC;
232
233     if(directCallableStackSizeFromTraversal)
234         *directCallableStackSizeFromTraversal = maxDCDepth * dssDC;
235     if(directCallableStackSizeFromState)
236         *directCallableStackSizeFromState = maxDCDepth * dssDC;
237
238     // upper bound on continuation stack used by CH or MS programs including the call tree of
239     // continuation callables
240     unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
241
242     // clang-format off
243     if(continuationStackSize)
244         *continuationStackSize
245             = cssRG + cssCCTree
246             + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
247             + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
248     // clang-format on
249
250     return OPTIX_SUCCESS;
251 }
252
268 inline OptixResult optixUtilComputeStackSizesSimplePathTracer(OptixProgramGroup        programGroupRG,
269                                                               OptixProgramGroup        programGroupMS1,
```

```
270                                                             const OptixProgramGroup* programGroupCH1,
271                                                       unsigned int            programGroupCH1Count,
272                                                             OptixProgramGroup       programGroupMS2,
273                                                             const OptixProgramGroup* programGroupCH2,
274                                                       unsigned int            programGroupCH2Count,
275                                                             unsigned int*
directCallableStackSizeFromTraversal,
276                                                     unsigned int* directCallableStackSizeFromState,
277                                                             unsigned int* continuationStackSize,
278                                                             OptixPipeline pipeline)
279 {
280     if(!programGroupCH1 && (programGroupCH1Count > 0))
281         return OPTIX_ERROR_INVALID_VALUE;
282     if(!programGroupCH2 && (programGroupCH2Count > 0))
283         return OPTIX_ERROR_INVALID_VALUE;
284
285     OptixResult result;
286
287     OptixStackSizes stackSizesRG = {};
288     result                       = optixProgramGroupGetStackSize(programGroupRG, &stackSizesRG, pipeline);
289     if(result != OPTIX_SUCCESS)
290         return result;
291
292     OptixStackSizes stackSizesMS1 = {};
293     result                        = optixProgramGroupGetStackSize(programGroupMS1, &stackSizesMS1,
pipeline);
294     if(result != OPTIX_SUCCESS)
295         return result;
296
297     OptixStackSizes stackSizesCH1 = {};
298     for(unsigned int i = 0; i < programGroupCH1Count; ++i)
299     {
300         result = optixUtilAccumulateStackSizes(programGroupCH1[i], &stackSizesCH1, pipeline);
301         if(result != OPTIX_SUCCESS)
302             return result;
303     }
304
305     OptixStackSizes stackSizesMS2 = {};
306     result                        = optixProgramGroupGetStackSize(programGroupMS2, &stackSizesMS2,
pipeline);
307     if(result != OPTIX_SUCCESS)
308         return result;
309
310     OptixStackSizes stackSizesCH2 = {};
311     memset(&stackSizesCH2, 0, sizeof(OptixStackSizes));
312     for(unsigned int i = 0; i < programGroupCH2Count; ++i)
313     {
314         result = optixUtilAccumulateStackSizes(programGroupCH2[i], &stackSizesCH2, pipeline);
315         if(result != OPTIX_SUCCESS)
316             return result;
317     }
318
319     const unsigned int cssRG  = stackSizesRG.cssRG;
320     const unsigned int cssMS1 = stackSizesMS1.cssMS;
321     const unsigned int cssCH1 = stackSizesCH1.cssCH;
322     const unsigned int cssMS2 = stackSizesMS2.cssMS;
323     const unsigned int cssCH2 = stackSizesCH2.cssCH;
324     // no AH, IS, CC, or DC programs
325
326     if(directCallableStackSizeFromTraversal)
327         *directCallableStackSizeFromTraversal = 0;
328     if(directCallableStackSizeFromState)
329         *directCallableStackSizeFromState = 0;
330
331     if(continuationStackSize)
332         *continuationStackSize = cssRG + std::max(cssMS1, cssCH1 + std::max(cssMS2, cssCH2));
333
```

```
334     return OPTIX_SUCCESS;
335 }
336   // end group optix_utilities
338
339 #ifdef __cplusplus
340 }
341 #endif
342
343 #endif  // OPTIX_OPTIX_STACK_SIZE_H
```

8.23 optix_stubs.h File Reference

Macros

- #define WIN32_LEAN_AND_MEAN 1

Functions

- static void * optixLoadWindowsDllFromName (const char *optixDllName)
- static void * optixLoadWindowsDll ()
- OptixResult optixInitWithHandle (void **handlePtr)
- OptixResult optixInit (void)
- OptixResult optixUninitWithHandle (void *handle)

Variables

- OptixFunctionTable g_optixFunctionTable

8.23.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.23.2 Macro Definition Documentation

8.23.2.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN 1
```

8.23.3 Function Documentation

8.23.3.1 optixLoadWindowsDll()

```
static void * optixLoadWindowsDll ( )  [static]
```

8.23.3.2 optixLoadWindowsDllFromName()

```
static void * optixLoadWindowsDllFromName (
          const char * optixDllName )  [static]
```

8.24 optix_stubs.h

Go to the documentation of this file.

```
1 /*
2 * Copyright (c) 2023 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 optixModuleCreate(OptixDeviceContext                 context,
388                                      const OptixModuleCompileOptions*   moduleCompileOptions,
389                                      const OptixPipelineCompileOptions* pipelineCompileOptions,
390                                      const char*                        input,
391                                      size_t                             inputSize,
392                                      char*                              logString,
393                                      size_t*                            logStringSize,
394                                      OptixModule*                       module)
395 {
396     return g_optixFunctionTable.optixModuleCreate(context, moduleCompileOptions, pipelineCompileOptions,
input, inputSize,
397                                                   logString, logStringSize, module);
398 }
399
400 inline OptixResult optixModuleCreateWithTasks(OptixDeviceContext                 context,
401                                               const OptixModuleCompileOptions*   moduleCompileOptions,
402                                               const OptixPipelineCompileOptions* pipelineCompileOptions,
403                                               const char*                        input,
404                                               size_t                             inputSize,
405                                               char*                              logString,
406                                               size_t*                            logStringSize,
407                                               OptixModule*                       module,
408                                               OptixTask*                         firstTask)
409 {
410     return g_optixFunctionTable.optixModuleCreateWithTasks(context, moduleCompileOptions,
pipelineCompileOptions, input,
411                                                            inputSize, logString, logStringSize, module,
firstTask);
```

```
412 }
413
414 inline OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState* state)
415 {
416     return g_optixFunctionTable.optixModuleGetCompilationState(module, state);
417 }
418
419 inline OptixResult optixModuleDestroy(OptixModule module)
420 {
421     return g_optixFunctionTable.optixModuleDestroy(module);
422 }
423
424 inline OptixResult optixBuiltinISModuleGet(OptixDeviceContext                 context,
425                                            const OptixModuleCompileOptions*   moduleCompileOptions,
426                                            const OptixPipelineCompileOptions* pipelineCompileOptions,
427                                            const OptixBuiltinISOptions*       builtinISOptions,
428                                            OptixModule*                       builtinModule)
429 {
430     return g_optixFunctionTable.optixBuiltinISModuleGet(context, moduleCompileOptions,
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, OptixPipeline pipeline)
457 {
458     return g_optixFunctionTable.optixProgramGroupGetStackSize(programGroup, stackSizes, pipeline);
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 optixAccelEmitProperty(OptixDeviceContext        context,
552                                           CUstream                  stream,
553                                           OptixTraversableHandle    handle,
554                                           const OptixAccelEmitDesc* emittedProperty)
555 {
556     return g_optixFunctionTable.optixAccelEmitProperty(context, stream, handle, emittedProperty);
557 }
558
559 inline OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext      onDevice,
560                                                           CUdeviceptr             pointer,
561                                                           OptixTraversableType    traversableType,
562                                                           OptixTraversableHandle* traversableHandle)
563 {
564     return g_optixFunctionTable.optixConvertPointerToTraversableHandle(onDevice, pointer,
traversableType, traversableHandle);
565 }
566
567 inline OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext
context,
568                                                                const OptixOpacityMicromapArrayBuildInput*
buildInput,
569                                                                  OptixMicromapBufferSizes*
bufferSizes)
570 {
571     return g_optixFunctionTable.optixOpacityMicromapArrayComputeMemoryUsage(context, buildInput,
bufferSizes);
572 }
573
574 inline OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext                       context,
575                                                   CUstream                                  stream,
576                                                   const OptixOpacityMicromapArrayBuildInput* buildInput,
577                                                   const OptixMicromapBuffers*               buffers)
578 {
579     return g_optixFunctionTable.optixOpacityMicromapArrayBuild(context, stream, buildInput, buffers);
580 }
581
582 inline OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext   context,
583                                                               CUdeviceptr          opacityMicromapArray,
584                                                               OptixRelocationInfo* info)
585 {
586     return g_optixFunctionTable.optixOpacityMicromapArrayGetRelocationInfo(context,
opacityMicromapArray, info);
587 }
588
589 inline OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext        context,
590                                                      CUstream                  stream,
591                                                      const OptixRelocationInfo* info,
592                                                      CUdeviceptr               targetOpacityMicromapArray,
593                                                      size_t
```

```
targetOpacityMicromapArraySizeInBytes)
594 {
595     return g_optixFunctionTable.optixOpacityMicromapArrayRelocate(context, stream, info,
targetOpacityMicromapArray, targetOpacityMicromapArraySizeInBytes);
596 }
597
598 inline OptixResult optixDisplacementMicromapArrayComputeMemoryUsage(OptixDeviceContext context,
599                                                                      const
OptixDisplacementMicromapArrayBuildInput* buildInput,
600                                                                    OptixMicromapBufferSizes* bufferSizes)
601 {
602     return g_optixFunctionTable.optixDisplacementMicromapArrayComputeMemoryUsage(context, buildInput,
bufferSizes);
603 }
604
605 inline OptixResult optixDisplacementMicromapArrayBuild(OptixDeviceContext
context,
606                                                        CUstream                                        stream,
607                                                            const OptixDisplacementMicromapArrayBuildInput*
buildInput,
608                                                       const OptixMicromapBuffers*                   buffers)
609 {
610     return g_optixFunctionTable.optixDisplacementMicromapArrayBuild(context, stream, buildInput,
buffers);
611 }
612
613 inline OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer)
614 {
615     return g_optixFunctionTable.optixSbtRecordPackHeader(programGroup, sbtRecordHeaderHostPointer);
616 }
617
618 inline OptixResult optixLaunch(OptixPipeline                 pipeline,
619                                CUstream                      stream,
620                                CUdeviceptr                   pipelineParams,
621                                size_t                        pipelineParamsSize,
622                                const OptixShaderBindingTable* sbt,
623                                unsigned int                  width,
624                                unsigned int                  height,
625                                unsigned int                  depth)
626 {
627     return g_optixFunctionTable.optixLaunch(pipeline, stream, pipelineParams, pipelineParamsSize, sbt,
width, height, depth);
628 }
629
630 inline OptixResult optixDenoiserCreate(OptixDeviceContext context, OptixDenoiserModelKind modelKind,
const OptixDenoiserOptions* options, OptixDenoiser* returnHandle)
631 {
632     return g_optixFunctionTable.optixDenoiserCreate(context, modelKind, options, returnHandle);
633 }
634
635 inline OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context, const void* data, size_t
dataSizeInBytes, OptixDenoiser* returnHandle)
636 {
637     return g_optixFunctionTable.optixDenoiserCreateWithUserModel(context, data, dataSizeInBytes,
returnHandle);
638 }
639
640 inline OptixResult optixDenoiserDestroy(OptixDenoiser handle)
641 {
642     return g_optixFunctionTable.optixDenoiserDestroy(handle);
643 }
644
645 inline OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser handle,
646                                                        unsigned int        maximumInputWidth,
647                                                        unsigned int        maximumInputHeight,
648                                                        OptixDenoiserSizes* returnSizes)
```

```
649 {
650     return g_optixFunctionTable.optixDenoiserComputeMemoryResources(handle, maximumInputWidth,
maximumInputHeight, returnSizes);
651 }
652
653 inline OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
654                                       CUstream      stream,
655                                       unsigned int  inputWidth,
656                                       unsigned int  inputHeight,
657                                       CUdeviceptr   denoiserState,
658                                       size_t        denoiserStateSizeInBytes,
659                                       CUdeviceptr   scratch,
660                                       size_t        scratchSizeInBytes)
661 {
662     return g_optixFunctionTable.optixDenoiserSetup(denoiser, stream, inputWidth, inputHeight,
denoiserState,
663                                                    denoiserStateSizeInBytes, scratch, scratchSizeInBytes);
664 }
665
666 inline OptixResult optixDenoiserInvoke(OptixDenoiser                   handle,
667                                        CUstream                        stream,
668                                        const OptixDenoiserParams*      params,
669                                        CUdeviceptr                     denoiserData,
670                                        size_t                          denoiserDataSize,
671                                        const OptixDenoiserGuideLayer*  guideLayer,
672                                        const OptixDenoiserLayer*       layers,
673                                        unsigned int                    numLayers,
674                                        unsigned int                    inputOffsetX,
675                                        unsigned int                    inputOffsetY,
676                                        CUdeviceptr                     scratch,
677                                        size_t                          scratchSizeInBytes)
678 {
679     return g_optixFunctionTable.optixDenoiserInvoke(handle, stream, params, denoiserData,
denoiserDataSize,
680                                                     guideLayer, layers, numLayers,
681                                                   inputOffsetX, inputOffsetY, scratch, scratchSizeInBytes);
682 }
683
684 inline OptixResult optixDenoiserComputeIntensity(OptixDenoiser       handle,
685                                                  CUstream            stream,
686                                                  const OptixImage2D* inputImage,
687                                                  CUdeviceptr         outputIntensity,
688                                                  CUdeviceptr         scratch,
689                                                  size_t              scratchSizeInBytes)
690 {
691     return g_optixFunctionTable.optixDenoiserComputeIntensity(handle, stream, inputImage,
outputIntensity, scratch, scratchSizeInBytes);
692 }
693
694 inline OptixResult optixDenoiserComputeAverageColor(OptixDenoiser       handle,
695                                                     CUstream            stream,
696                                                     const OptixImage2D* inputImage,
697                                                     CUdeviceptr         outputAverageColor,
698                                                     CUdeviceptr         scratch,
699                                                     size_t              scratchSizeInBytes)
700 {
701     return g_optixFunctionTable.optixDenoiserComputeAverageColor(handle, stream, inputImage,
outputAverageColor, scratch, scratchSizeInBytes);
702 }
703
704 #endif  // OPTIX_DOXYGEN_SHOULD_SKIP_THIS
705
706 #ifdef __cplusplus
707 }
708 #endif
709
710 #endif  // OPTIX_OPTIX_STUBS_H
```

8.25 optix_types.h File Reference

Classes

- struct OptixDeviceContextOptions
- struct OptixOpacityMicromapUsageCount
- struct OptixBuildInputOpacityMicromap
- struct OptixRelocateInputOpacityMicromap
- struct OptixDisplacementMicromapDesc
- struct OptixDisplacementMicromapHistogramEntry
- struct OptixDisplacementMicromapArrayBuildInput
- struct OptixDisplacementMicromapUsageCount
- struct OptixBuildInputDisplacementMicromap
- struct OptixBuildInputTriangleArray
- struct OptixRelocateInputTriangleArray
- struct OptixBuildInputCurveArray
- struct OptixBuildInputSphereArray
- struct OptixAabb
- struct OptixBuildInputCustomPrimitiveArray
- struct OptixBuildInputInstanceArray
- struct OptixRelocateInputInstanceArray
- struct OptixBuildInput
- struct OptixRelocateInput
- struct OptixInstance
- struct OptixOpacityMicromapDesc
- struct OptixOpacityMicromapHistogramEntry
- struct OptixOpacityMicromapArrayBuildInput
- struct OptixMicromapBufferSizes
- struct OptixMicromapBuffers
- struct OptixMotionOptions
- struct OptixAccelBuildOptions
- struct OptixAccelBufferSizes
- struct OptixAccelEmitDesc
- struct OptixRelocationInfo
- struct OptixStaticTransform
- struct OptixMatrixMotionTransform
- struct OptixSRTData
- struct OptixSRTMotionTransform
- struct OptixImage2D
- struct OptixDenoiserOptions
- struct OptixDenoiserGuideLayer
- struct OptixDenoiserLayer
- struct OptixDenoiserParams
- struct OptixDenoiserSizes
- struct OptixModuleCompileBoundValueEntry
- struct OptixPayloadType
- struct OptixModuleCompileOptions
- struct OptixProgramGroupSingleModule
- struct OptixProgramGroupHitgroup
- struct OptixProgramGroupCallables
- struct OptixProgramGroupDesc

- struct OptixProgramGroupOptions
- struct OptixPipelineCompileOptions
- struct OptixPipelineLinkOptions
- struct OptixShaderBindingTable
- struct OptixStackSizes
- struct 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_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
- #define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
- #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT (0)
- #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE (1)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT (2)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE (3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT (-1)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE (-2)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT (-3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE (-4)
- #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
- #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
- #define OPTIX_DISPLACEMENT_MICROMAP_MAX_SUBDIVISION_LEVEL 5
- #define OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
- #define OPTIX_DISPLACEMENT_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull

Typedefs

- typedef unsigned long long CUdeviceptr
- typedef struct OptixDeviceContext_t * OptixDeviceContext
- typedef struct OptixModule_t * OptixModule
- typedef struct OptixProgramGroup_t * OptixProgramGroup
- typedef struct OptixPipeline_t * OptixPipeline
- typedef struct OptixDenoiser_t * OptixDenoiser
- typedef struct OptixTask_t * OptixTask
- typedef unsigned long long OptixTraversableHandle
- typedef unsigned int OptixVisibilityMask
- typedef enum OptixResult OptixResult
- typedef enum OptixDeviceProperty OptixDeviceProperty
- typedef void(* OptixLogCallback) (unsigned int level, const char *tag, const char *message, void *cbdata)

- typedef enum OptixDeviceContextValidationMode OptixDeviceContextValidationMode
- typedef struct OptixDeviceContextOptions OptixDeviceContextOptions
- typedef enum OptixDevicePropertyShaderExecutionReorderingFlags OptixDevicePropertyShaderExecutionReorderingFlags
- typedef enum OptixGeometryFlags OptixGeometryFlags
- typedef enum OptixHitKind OptixHitKind
- typedef enum OptixIndicesFormat OptixIndicesFormat
- typedef enum OptixVertexFormat OptixVertexFormat
- typedef enum OptixTransformFormat OptixTransformFormat
- typedef enum OptixDisplacementMicromapBiasAndScaleFormat OptixDisplacementMicromapBiasAndScaleFormat
- typedef enum OptixDisplacementMicromapDirectionFormat OptixDisplacementMicromapDirectionFormat
- typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat
- typedef enum OptixOpacityMicromapArrayIndexingMode OptixOpacityMicromapArrayIndexingMode
- typedef struct OptixOpacityMicromapUsageCount OptixOpacityMicromapUsageCount
- typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap
- typedef struct OptixRelocateInputOpacityMicromap OptixRelocateInputOpacityMicromap
- typedef enum OptixDisplacementMicromapFormat OptixDisplacementMicromapFormat
- typedef enum OptixDisplacementMicromapFlags OptixDisplacementMicromapFlags
- typedef enum OptixDisplacementMicromapTriangleFlags OptixDisplacementMicromapTriangleFlags
- typedef struct OptixDisplacementMicromapDesc OptixDisplacementMicromapDesc
- typedef struct OptixDisplacementMicromapHistogramEntry OptixDisplacementMicromapHistogramEntry
- typedef struct OptixDisplacementMicromapArrayBuildInput OptixDisplacementMicromapArrayBuildInput
- typedef struct OptixDisplacementMicromapUsageCount OptixDisplacementMicromapUsageCount
- typedef enum OptixDisplacementMicromapArrayIndexingMode OptixDisplacementMicromapArrayIndexingMode
- typedef struct OptixBuildInputDisplacementMicromap OptixBuildInputDisplacementMicromap
- typedef struct OptixBuildInputTriangleArray OptixBuildInputTriangleArray
- typedef struct OptixRelocateInputTriangleArray OptixRelocateInputTriangleArray
- typedef enum OptixPrimitiveType OptixPrimitiveType
- typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags
- typedef enum OptixCurveEndcapFlags OptixCurveEndcapFlags
- typedef struct OptixBuildInputCurveArray OptixBuildInputCurveArray
- typedef struct OptixBuildInputSphereArray OptixBuildInputSphereArray
- typedef struct OptixAabb OptixAabb
- typedef struct OptixBuildInputCustomPrimitiveArray OptixBuildInputCustomPrimitiveArray
- typedef struct OptixBuildInputInstanceArray OptixBuildInputInstanceArray
- typedef struct OptixRelocateInputInstanceArray OptixRelocateInputInstanceArray
- typedef enum OptixBuildInputType OptixBuildInputType
- typedef struct OptixBuildInput OptixBuildInput
- typedef struct OptixRelocateInput OptixRelocateInput
- typedef enum OptixInstanceFlags OptixInstanceFlags
- typedef struct OptixInstance OptixInstance
- typedef enum OptixBuildFlags OptixBuildFlags

- typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags
- typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc
- typedef struct OptixOpacityMicromapHistogramEntry OptixOpacityMicromapHistogramEntry
- typedef struct OptixOpacityMicromapArrayBuildInput OptixOpacityMicromapArrayBuildInput
- typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
- typedef struct OptixMicromapBuffers OptixMicromapBuffers
- typedef enum OptixBuildOperation OptixBuildOperation
- typedef enum OptixMotionFlags OptixMotionFlags
- typedef struct OptixMotionOptions OptixMotionOptions
- typedef struct OptixAccelBuildOptions OptixAccelBuildOptions
- typedef struct OptixAccelBufferSizes OptixAccelBufferSizes
- typedef enum OptixAccelPropertyType OptixAccelPropertyType
- typedef struct OptixAccelEmitDesc OptixAccelEmitDesc
- typedef struct OptixRelocationInfo OptixRelocationInfo
- typedef struct OptixStaticTransform OptixStaticTransform
- typedef struct OptixMatrixMotionTransform OptixMatrixMotionTransform
- typedef struct OptixSRTData OptixSRTData
- typedef struct OptixSRTMotionTransform OptixSRTMotionTransform
- typedef enum OptixTraversableType OptixTraversableType
- typedef enum OptixPixelFormat OptixPixelFormat
- typedef struct OptixImage2D OptixImage2D
- typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
- typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
- typedef struct OptixDenoiserOptions OptixDenoiserOptions
- typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
- typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
- typedef struct OptixDenoiserLayer OptixDenoiserLayer
- typedef struct OptixDenoiserParams OptixDenoiserParams
- typedef struct OptixDenoiserSizes OptixDenoiserSizes
- typedef enum OptixRayFlags OptixRayFlags
- typedef enum OptixTransformType OptixTransformType
- typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags
- typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
- typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
- typedef enum OptixModuleCompileState OptixModuleCompileState
- typedef struct OptixModuleCompileBoundValueEntry OptixModuleCompileBoundValueEntry
- typedef enum OptixPayloadTypeID OptixPayloadTypeID
- typedef enum OptixPayloadSemantics OptixPayloadSemantics
- typedef struct OptixPayloadType OptixPayloadType
- typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
- typedef enum OptixProgramGroupKind OptixProgramGroupKind
- typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
- typedef struct OptixProgramGroupSingleModule OptixProgramGroupSingleModule
- typedef struct OptixProgramGroupHitgroup OptixProgramGroupHitgroup
- typedef struct OptixProgramGroupCallables OptixProgramGroupCallables
- typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
- typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
- typedef enum OptixExceptionCodes OptixExceptionCodes
- typedef enum OptixExceptionFlags OptixExceptionFlags
- typedef struct OptixPipelineCompileOptions OptixPipelineCompileOptions

- typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
- typedef struct OptixShaderBindingTable OptixShaderBindingTable
- typedef struct OptixStackSizes OptixStackSizes
- typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
- typedef OptixResult() OptixQueryFunctionTable_t(int abiId, unsigned int numOptions, OptixQueryFunctionTableOptions *, const void **, void *functionTable, size_t sizeOfTable)
- typedef struct 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_INPUT = 7200 ,
  OPTIX_ERROR_INVALID_LAUNCH_PARAMETER = 7201 ,
  OPTIX_ERROR_INVALID_PAYLOAD_ACCESS = 7202 ,
  OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS = 7203 ,
  OPTIX_ERROR_INVALID_FUNCTION_USE = 7204 ,
  OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS = 7205 ,
  OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250 ,
  OPTIX_ERROR_PIPELINE_LINK_ERROR = 7251 ,
  OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE = 7270 ,
  OPTIX_ERROR_INTERNAL_COMPILER_ERROR = 7299 ,
  OPTIX_ERROR_DENOISER_MODEL_NOT_SET = 7300 ,
  OPTIX_ERROR_DENOISER_NOT_INITIALIZED = 7301 ,
  OPTIX_ERROR_NOT_COMPATIBLE = 7400 ,
  OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH = 7500 ,
  OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED = 7501 ,
  OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID = 7502 ,
  OPTIX_ERROR_NOT_SUPPORTED = 7800 ,
  OPTIX_ERROR_UNSUPPORTED_ABI_VERSION = 7801 ,
  OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH = 7802 ,
  OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS = 7803 ,
  OPTIX_ERROR_LIBRARY_NOT_FOUND = 7804 ,
  OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND = 7805 ,
  OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE = 7806 ,
  OPTIX_ERROR_DEVICE_OUT_OF_MEMORY = 7807 ,
  OPTIX_ERROR_CUDA_ERROR = 7900 ,
  OPTIX_ERROR_INTERNAL_ERROR = 7990 ,
  OPTIX_ERROR_UNKNOWN = 7999 }
- enum OptixDeviceProperty {
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001 ,

OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002 ,
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003 ,
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004 ,
OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005 ,
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006 ,
OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007 ,
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008 ,
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009 ,
OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING = 0x200A }

- enum OptixDeviceContextValidationMode {
  OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0 ,
  OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF }
- enum OptixDevicePropertyShaderExecutionReorderingFlags {
  OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_NONE = 0 ,
  OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD = 1 << 0 }
- enum OptixGeometryFlags {
  OPTIX_GEOMETRY_FLAG_NONE = 0 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
  OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u << 1 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 2 }
- enum OptixHitKind {
  OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE ,
  OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF }
- enum OptixIndicesFormat {
  OPTIX_INDICES_FORMAT_NONE = 0 ,
  OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102 ,
  OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103 }
- enum OptixVertexFormat {
  OPTIX_VERTEX_FORMAT_NONE = 0 ,
  OPTIX_VERTEX_FORMAT_FLOAT3 = 0x2121 ,
  OPTIX_VERTEX_FORMAT_FLOAT2 = 0x2122 ,
  OPTIX_VERTEX_FORMAT_HALF3 = 0x2123 ,
  OPTIX_VERTEX_FORMAT_HALF2 = 0x2124 ,
  OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125 ,
  OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126 }
- enum OptixTransformFormat {
  OPTIX_TRANSFORM_FORMAT_NONE = 0 ,
  OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1 }
- enum OptixDisplacementMicromapBiasAndScaleFormat {
  OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_NONE = 0 ,
  OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_FLOAT2 = 0x2241 ,
  OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_HALF2 = 0x2242 }
- enum OptixDisplacementMicromapDirectionFormat {
  OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_NONE = 0 ,
  OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_FLOAT3 = 0x2261 ,
  OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_HALF3 = 0x2262 }
- enum OptixOpacityMicromapFormat {
  OPTIX_OPACITY_MICROMAP_FORMAT_NONE = 0 ,
  OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE = 1 ,
  OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE = 2 }
- enum OptixOpacityMicromapArrayIndexingMode {
  OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0 ,

OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1 ,
OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2 }

- enum OptixDisplacementMicromapFormat {
OPTIX_DISPLACEMENT_MICROMAP_FORMAT_NONE = 0 ,
OPTIX_DISPLACEMENT_MICROMAP_FORMAT_64_MICRO_TRIS_64_BYTES = 1 ,
OPTIX_DISPLACEMENT_MICROMAP_FORMAT_256_MICRO_TRIS_128_BYTES = 2 ,
OPTIX_DISPLACEMENT_MICROMAP_FORMAT_1024_MICRO_TRIS_128_BYTES = 3 }
- enum OptixDisplacementMicromapFlags {
OPTIX_DISPLACEMENT_MICROMAP_FLAG_NONE = 0 ,
OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 << 0 ,
OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 << 1 }
- enum OptixDisplacementMicromapTriangleFlags {
OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_NONE = 0 ,
OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_01 = 1 << 0 ,
OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_12 = 1 << 1 ,
OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_20 = 1 << 2 }
- enum OptixDisplacementMicromapArrayIndexingMode {
OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0 ,
OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1 ,
OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2 }
- enum OptixPrimitiveType {
OPTIX_PRIMITIVE_TYPE_CUSTOM = 0x2500 ,
OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE = 0x2501 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE = 0x2502 ,
OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR = 0x2503 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM = 0x2504 ,
OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE = 0x2505 ,
OPTIX_PRIMITIVE_TYPE_SPHERE = 0x2506 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER = 0x2507 ,
OPTIX_PRIMITIVE_TYPE_TRIANGLE = 0x2531 ,
OPTIX_PRIMITIVE_TYPE_DISPLACED_MICROMESH_TRIANGLE = 0x2532 }
- enum OptixPrimitiveTypeFlags {
OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM = 1 << 0 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE = 1 << 1 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE = 1 << 2 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR = 1 << 3 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM = 1 << 4 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE = 1 << 5 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE = 1 << 6 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER = 1 << 7 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE = 1 << 31 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_DISPLACED_MICROMESH_TRIANGLE = 1 << 30 }
- enum OptixCurveEndcapFlags {
OPTIX_CURVE_ENDCAP_DEFAULT = 0 ,
OPTIX_CURVE_ENDCAP_ON = 1 << 0 }
- enum OptixBuildInputType {
OPTIX_BUILD_INPUT_TYPE_TRIANGLES = 0x2141 ,
OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES = 0x2142 ,
OPTIX_BUILD_INPUT_TYPE_INSTANCES = 0x2143 ,
OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS = 0x2144 ,
OPTIX_BUILD_INPUT_TYPE_CURVES = 0x2145 ,
OPTIX_BUILD_INPUT_TYPE_SPHERES = 0x2146 }

- enum OptixInstanceFlags {
  OPTIX_INSTANCE_FLAG_NONE = 0 ,
  OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 0 ,
  OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u << 1 ,
  OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT = 1u << 2 ,
  OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u << 3 ,
  OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 4 ,
  OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS = 1u << 5 }
- enum OptixBuildFlags {
  OPTIX_BUILD_FLAG_NONE = 0 ,
  OPTIX_BUILD_FLAG_ALLOW_UPDATE = 1u << 0 ,
  OPTIX_BUILD_FLAG_ALLOW_COMPACTION = 1u << 1 ,
  OPTIX_BUILD_FLAG_PREFER_FAST_TRACE = 1u << 2 ,
  OPTIX_BUILD_FLAG_PREFER_FAST_BUILD = 1u << 3 ,
  OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS = 1u << 4 ,
  OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS = 1u << 5 ,
  OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE = 1u << 6 ,
  OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1u << 7 }
- enum OptixOpacityMicromapFlags {
  OPTIX_OPACITY_MICROMAP_FLAG_NONE = 0 ,
  OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 << 0 ,
  OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 << 1 }
- enum OptixBuildOperation {
  OPTIX_BUILD_OPERATION_BUILD = 0x2161 ,
  OPTIX_BUILD_OPERATION_UPDATE = 0x2162 }
- enum OptixMotionFlags {
  OPTIX_MOTION_FLAG_NONE = 0 ,
  OPTIX_MOTION_FLAG_START_VANISH = 1u << 0 ,
  OPTIX_MOTION_FLAG_END_VANISH = 1u << 1 }
- enum OptixAccelPropertyType {
  OPTIX_PROPERTY_TYPE_COMPACTED_SIZE = 0x2181 ,
  OPTIX_PROPERTY_TYPE_AABBS = 0x2182 }
- enum OptixTraversableType {
  OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM = 0x21C1 ,
  OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM = 0x21C2 ,
  OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM = 0x21C3 }
- enum OptixPixelFormat {
  OPTIX_PIXEL_FORMAT_HALF1 = 0x220a ,
  OPTIX_PIXEL_FORMAT_HALF2 = 0x2207 ,
  OPTIX_PIXEL_FORMAT_HALF3 = 0x2201 ,
  OPTIX_PIXEL_FORMAT_HALF4 = 0x2202 ,
  OPTIX_PIXEL_FORMAT_FLOAT1 = 0x220b ,
  OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208 ,
  OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203 ,
  OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204 ,
  OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205 ,
  OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206 ,
  OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209 }
- enum OptixDenoiserModelKind {
  OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322 ,
  OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323 ,
  OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325 ,

OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326 ,
OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327 ,
OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328 }

- enum OptixDenoiserAlphaMode {
  OPTIX_DENOISER_ALPHA_MODE_COPY = 0 ,
  OPTIX_DENOISER_ALPHA_MODE_DENOISE = 1 }
- enum OptixDenoiserAOVType {
  OPTIX_DENOISER_AOV_TYPE_NONE = 0 ,
  OPTIX_DENOISER_AOV_TYPE_BEAUTY = 0x7000 ,
  OPTIX_DENOISER_AOV_TYPE_SPECULAR = 0x7001 ,
  OPTIX_DENOISER_AOV_TYPE_REFLECTION = 0x7002 ,
  OPTIX_DENOISER_AOV_TYPE_REFRACTION = 0x7003 ,
  OPTIX_DENOISER_AOV_TYPE_DIFFUSE = 0x7004 }
- enum OptixRayFlags {
  OPTIX_RAY_FLAG_NONE = 0u ,
  OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
  OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u << 1 ,
  OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u << 2 ,
  OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u << 3 ,
  OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u << 4 ,
  OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u << 5 ,
  OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u << 6 ,
  OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u << 7 ,
  OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 10 }
- enum OptixTransformType {
  OPTIX_TRANSFORM_TYPE_NONE = 0 ,
  OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM = 1 ,
  OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM = 2 ,
  OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM = 3 ,
  OPTIX_TRANSFORM_TYPE_INSTANCE = 4 }
- enum OptixTraversableGraphFlags {
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u << 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u << 1 }
- enum OptixCompileOptimizationLevel {
  OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343 }
- enum OptixCompileDebugLevel {
  OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT = 0 ,
  OPTIX_COMPILE_DEBUG_LEVEL_NONE = 0x2350 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL = 0x2351 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353 ,
  OPTIX_COMPILE_DEBUG_LEVEL_FULL = 0x2352 }
- enum OptixModuleCompileState {
  OPTIX_MODULE_COMPILE_STATE_NOT_STARTED = 0x2360 ,
  OPTIX_MODULE_COMPILE_STATE_STARTED = 0x2361 ,
  OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE = 0x2362 ,
  OPTIX_MODULE_COMPILE_STATE_FAILED = 0x2363 ,
  OPTIX_MODULE_COMPILE_STATE_COMPLETED = 0x2364 }

- enum OptixPayloadTypeID {
  OPTIX_PAYLOAD_TYPE_DEFAULT = 0 ,
  OPTIX_PAYLOAD_TYPE_ID_0 = (1 << 0u) ,
  OPTIX_PAYLOAD_TYPE_ID_1 = (1 << 1u) ,
  OPTIX_PAYLOAD_TYPE_ID_2 = (1 << 2u) ,
  OPTIX_PAYLOAD_TYPE_ID_3 = (1 << 3u) ,
  OPTIX_PAYLOAD_TYPE_ID_4 = (1 << 4u) ,
  OPTIX_PAYLOAD_TYPE_ID_5 = (1 << 5u) ,
  OPTIX_PAYLOAD_TYPE_ID_6 = (1 << 6u) ,
  OPTIX_PAYLOAD_TYPE_ID_7 = (1 << 7u) }
- enum OptixPayloadSemantics {
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ = 1u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE = 2u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_READ = 1u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_WRITE = 2u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE = 3u << 2 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_READ = 1u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_WRITE = 2u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE = 3u << 4 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_READ = 1u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_WRITE = 2u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE = 3u << 6 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_READ = 1u << 8 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_WRITE = 2u << 8 ,
  OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE = 3u << 8 }
- enum OptixProgramGroupKind {
  OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421 ,
  OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422 ,
  OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423 ,
  OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424 ,
  OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425 }
- enum OptixProgramGroupFlags { OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0 }
- enum OptixExceptionCodes {
  OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1 ,
  OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2 }
- enum OptixExceptionFlags {
  OPTIX_EXCEPTION_FLAG_NONE = 0 ,
  OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u << 0 ,
  OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u << 1 ,
  OPTIX_EXCEPTION_FLAG_USER = 1u << 2 }
- enum OptixQueryFunctionTableOptions { OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0 }

8.25.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

OptiX types include file – defines types and enums used by the API. For the math library routines include optix_math.h

8.26 optix_types.h

Go to the documentation of this file.

```
1
2 /*
3 * Copyright (c) 2023 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_TYPES_H
30 #define OPTIX_OPTIX_TYPES_H
31
32 #if !defined(__CUDACC_RTC__)
33 #include <stddef.h> /* for size_t */
34 #endif
35
36 #ifdef NV_MODULE_OPTIX
37 // This is a mechanism to include <g_nvconfig.h> in driver builds only and translate any nvconfig macro to
a custom OPTIX-specific macro, that can also be used in SDK builds/installs
38 #include <exp/misc/optix_nvconfig_translate.h>  // includes <g_nvconfig.h>
39 #endif  // NV_MODULE_OPTIX
40
41
44
49 // This typedef should match the one in cuda.h in order to avoid compilation errors.
50 #if defined(_WIN64) || defined(__LP64__)
52 typedef unsigned long long CUdeviceptr;
53 #else
55 typedef unsigned int CUdeviceptr;
56 #endif
57
59 typedef struct OptixDeviceContext_t* OptixDeviceContext;
60
62 typedef struct OptixModule_t* OptixModule;
63
65 typedef struct OptixProgramGroup_t* OptixProgramGroup;
66
68 typedef struct OptixPipeline_t* OptixPipeline;
69
71 typedef struct OptixDenoiser_t* OptixDenoiser;
72
74 typedef struct OptixTask_t* OptixTask;
75
```

```
77 typedef unsigned long long OptixTraversableHandle;
78
80 typedef unsigned int OptixVisibilityMask;
81
83 #define OPTIX_SBT_RECORD_HEADER_SIZE ((size_t)32)
84
86 #define OPTIX_SBT_RECORD_ALIGNMENT 16ull
87
89 #define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128ull
90
92 #define OPTIX_INSTANCE_BYTE_ALIGNMENT 16ull
93
95 #define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8ull
96
98 #define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16ull
99
101 #define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
102
104 #define OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
105
107 #define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
108
110 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
111
113 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
114
117 #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT          (0)
118 #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE               (1)
119 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT  (2)
120 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE       (3)
121
124 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT          (-1)
125 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE               (-2)
126 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT  (-3)
127 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE       (-4)
128
130 #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
131
133 #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
134
136 #define OPTIX_DISPLACEMENT_MICROMAP_MAX_SUBDIVISION_LEVEL 5
137
139 #define OPTIX_DISPLACEMENT_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
140
142 #define OPTIX_DISPLACEMENT_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
143
151 typedef enum OptixResult
152 {
153     OPTIX_SUCCESS                                = 0,
154     OPTIX_ERROR_INVALID_VALUE                    = 7001,
155     OPTIX_ERROR_HOST_OUT_OF_MEMORY               = 7002,
156     OPTIX_ERROR_INVALID_OPERATION                = 7003,
157     OPTIX_ERROR_FILE_IO_ERROR                    = 7004,
158     OPTIX_ERROR_INVALID_FILE_FORMAT              = 7005,
159     OPTIX_ERROR_DISK_CACHE_INVALID_PATH          = 7010,
160     OPTIX_ERROR_DISK_CACHE_PERMISSION_ERROR      = 7011,
161     OPTIX_ERROR_DISK_CACHE_DATABASE_ERROR        = 7012,
162     OPTIX_ERROR_DISK_CACHE_INVALID_DATA          = 7013,
163     OPTIX_ERROR_LAUNCH_FAILURE                   = 7050,
164     OPTIX_ERROR_INVALID_DEVICE_CONTEXT           = 7051,
165     OPTIX_ERROR_CUDA_NOT_INITIALIZED             = 7052,
166     OPTIX_ERROR_VALIDATION_FAILURE               = 7053,
167     OPTIX_ERROR_INVALID_INPUT                    = 7200,
168     OPTIX_ERROR_INVALID_LAUNCH_PARAMETER         = 7201,
169     OPTIX_ERROR_INVALID_PAYLOAD_ACCESS           = 7202,
170     OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS         = 7203,
171     OPTIX_ERROR_INVALID_FUNCTION_USE             = 7204,
```

```
172     OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS      = 7205,
173     OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250,
174     OPTIX_ERROR_PIPELINE_LINK_ERROR             = 7251,
175     OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE     = 7270,
176     OPTIX_ERROR_INTERNAL_COMPILER_ERROR         = 7299,
177     OPTIX_ERROR_DENOISER_MODEL_NOT_SET          = 7300,
178     OPTIX_ERROR_DENOISER_NOT_INITIALIZED        = 7301,
179     OPTIX_ERROR_NOT_COMPATIBLE                  = 7400,
180     OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH           = 7500,
181     OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED  = 7501,
182     OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID         = 7502,
183     OPTIX_ERROR_NOT_SUPPORTED                   = 7800,
184     OPTIX_ERROR_UNSUPPORTED_ABI_VERSION         = 7801,
185     OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH    = 7802,
186     OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS  = 7803,
187     OPTIX_ERROR_LIBRARY_NOT_FOUND               = 7804,
188     OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND          = 7805,
189     OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE          = 7806,
190     OPTIX_ERROR_DEVICE_OUT_OF_MEMORY            = 7807,
191     OPTIX_ERROR_CUDA_ERROR                      = 7900,
192     OPTIX_ERROR_INTERNAL_ERROR                  = 7990,
193     OPTIX_ERROR_UNKNOWN                         = 7999,
194 } OptixResult;
195
199 typedef enum OptixDeviceProperty
200 {
202     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001,
203
206     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002,
207
210     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003,
211
214     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004,
215
218     OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005,
219
221     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006,
222
225     OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007,
226
229     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008,
230
234     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009,
235
239     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING = 0x200A,
240 } OptixDeviceProperty;
241
266 typedef void (*OptixLogCallback)(unsigned int level, const char* tag, const char* message, void* cbdata);
267
275 typedef enum OptixDeviceContextValidationMode
276 {
277     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0,
278     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF
279 } OptixDeviceContextValidationMode;
280
284 typedef struct OptixDeviceContextOptions
285 {
287     OptixLogCallback logCallbackFunction;
289     void* logCallbackData;
291     int logCallbackLevel;
293     OptixDeviceContextValidationMode validationMode;
294 } OptixDeviceContextOptions;
295
300 typedef enum OptixDevicePropertyShaderExecutionReorderingFlags
301 {
304     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_NONE     = 0,
305
```

```
306     // Standard thread reordering is supported
307     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD = 1 « 0,
308 } OptixDevicePropertyShaderExecutionReorderingFlags;
309
312 typedef enum OptixGeometryFlags
313 {
315     OPTIX_GEOMETRY_FLAG_NONE = 0,
316
319     OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u « 0,
320
324     OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u « 1,
325
329     OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u « 2,
330 } OptixGeometryFlags;
331
337 typedef enum OptixHitKind
338 {
340     OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE,
342     OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF
343 } OptixHitKind;
344
346 typedef enum OptixIndicesFormat
347 {
349     OPTIX_INDICES_FORMAT_NONE = 0,
351     OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102,
353     OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103
354 } OptixIndicesFormat;
355
357 typedef enum OptixVertexFormat
358 {
359     OPTIX_VERTEX_FORMAT_NONE      = 0,
360     OPTIX_VERTEX_FORMAT_FLOAT3    = 0x2121,
361     OPTIX_VERTEX_FORMAT_FLOAT2    = 0x2122,
362     OPTIX_VERTEX_FORMAT_HALF3     = 0x2123,
363     OPTIX_VERTEX_FORMAT_HALF2     = 0x2124,
364     OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125,
365     OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126
366 } OptixVertexFormat;
367
369 typedef enum OptixTransformFormat
370 {
371     OPTIX_TRANSFORM_FORMAT_NONE           = 0,
372     OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1,
373 } OptixTransformFormat;
374
375 typedef enum OptixDisplacementMicromapBiasAndScaleFormat
376 {
377     OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_NONE   = 0,
378     OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_FLOAT2 = 0x2241,
379     OPTIX_DISPLACEMENT_MICROMAP_BIAS_AND_SCALE_FORMAT_HALF2  = 0x2242,
380 } OptixDisplacementMicromapBiasAndScaleFormat;
381
382 typedef enum OptixDisplacementMicromapDirectionFormat
383 {
384     OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_NONE   = 0,
385     OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_FLOAT3 = 0x2261,
386     OPTIX_DISPLACEMENT_MICROMAP_DIRECTION_FORMAT_HALF3  = 0x2262,
387 } OptixDisplacementMicromapDirectionFormat;
388
390 typedef enum OptixOpacityMicromapFormat
391 {
393     OPTIX_OPACITY_MICROMAP_FORMAT_NONE = 0,
395     OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE = 1,
397     OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE = 2,
398 } OptixOpacityMicromapFormat;
399
401 typedef enum OptixOpacityMicromapArrayIndexingMode
```

```
402 {
404     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0,
407     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1,
411     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2,
412 } OptixOpacityMicromapArrayIndexingMode;
413
418 typedef struct OptixOpacityMicromapUsageCount
419 {
422     unsigned int count;
424     unsigned int subdivisionLevel;
426     OptixOpacityMicromapFormat format;
427 } OptixOpacityMicromapUsageCount;
428
429 typedef struct OptixBuildInputOpacityMicromap
430 {
432     OptixOpacityMicromapArrayIndexingMode indexingMode;
433
438     CUdeviceptr  opacityMicromapArray;
439
449     CUdeviceptr  indexBuffer;
450
453     unsigned int indexSizeInBytes;
454
457     unsigned int indexStrideInBytes;
458
460     unsigned int indexOffset;
461
463     unsigned int numMicromapUsageCounts;
466     const OptixOpacityMicromapUsageCount* micromapUsageCounts;
467 } OptixBuildInputOpacityMicromap;
468
469 typedef struct OptixRelocateInputOpacityMicromap
470 {
474     CUdeviceptr  opacityMicromapArray;
475 } OptixRelocateInputOpacityMicromap;
476
477
479 typedef enum OptixDisplacementMicromapFormat
480 {
481     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_NONE                       = 0,
482     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_64_MICRO_TRIS_64_BYTES     = 1,
483     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_256_MICRO_TRIS_128_BYTES   = 2,
484     OPTIX_DISPLACEMENT_MICROMAP_FORMAT_1024_MICRO_TRIS_128_BYTES = 3,
485 } OptixDisplacementMicromapFormat;
486
488 typedef enum OptixDisplacementMicromapFlags
489 {
490     OPTIX_DISPLACEMENT_MICROMAP_FLAG_NONE = 0,
491
493     OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 « 0,
494
496     OPTIX_DISPLACEMENT_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 « 1,
497
498 } OptixDisplacementMicromapFlags;
499
500 typedef enum OptixDisplacementMicromapTriangleFlags
501 {
502     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_NONE             = 0,
505     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_01 = 1 « 0,
507     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_12 = 1 « 1,
509     OPTIX_DISPLACEMENT_MICROMAP_TRIANGLE_FLAG_DECIMATE_EDGE_20 = 1 « 2,
510 } OptixDisplacementMicromapTriangleFlags;
511
512 typedef struct OptixDisplacementMicromapDesc
513 {
515     unsigned int   byteOffset;
517     unsigned short subdivisionLevel;
```

```
519     unsigned short format;
520 } OptixDisplacementMicromapDesc;
521
526 typedef struct OptixDisplacementMicromapHistogramEntry
527 {
529     unsigned int                    count;
531     unsigned int                    subdivisionLevel;
533     OptixDisplacementMicromapFormat format;
534 } OptixDisplacementMicromapHistogramEntry;
535
537 typedef struct OptixDisplacementMicromapArrayBuildInput
538 {
540     OptixDisplacementMicromapFlags                flags;
542     CUdeviceptr                                   displacementValuesBuffer;
545     CUdeviceptr                                   perDisplacementMicromapDescBuffer;
549     unsigned int                                  perDisplacementMicromapDescStrideInBytes;
551     unsigned int                                  numDisplacementMicromapHistogramEntries;
554     const OptixDisplacementMicromapHistogramEntry* displacementMicromapHistogramEntries;
555 } OptixDisplacementMicromapArrayBuildInput;
556
561 typedef struct OptixDisplacementMicromapUsageCount
562 {
565     unsigned int                    count;
567     unsigned int                    subdivisionLevel;
569     OptixDisplacementMicromapFormat format;
570 } OptixDisplacementMicromapUsageCount;
571
573 typedef enum OptixDisplacementMicromapArrayIndexingMode
574 {
576     OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0,
579     OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1,
583     OPTIX_DISPLACEMENT_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2,
584 } OptixDisplacementMicromapArrayIndexingMode;
585
587 typedef struct OptixBuildInputDisplacementMicromap
588 {
590     OptixDisplacementMicromapArrayIndexingMode indexingMode;
591
593     CUdeviceptr displacementMicromapArray;
595     CUdeviceptr displacementMicromapIndexBuffer;
597     CUdeviceptr vertexDirectionsBuffer;
599     CUdeviceptr vertexBiasAndScaleBuffer;
601     CUdeviceptr triangleFlagsBuffer;
602
604     unsigned int displacementMicromapIndexOffset;
607     unsigned int displacementMicromapIndexStrideInBytes;
609     unsigned int displacementMicromapIndexSizeInBytes;
610
612     OptixDisplacementMicromapDirectionFormat vertexDirectionFormat;
614     unsigned int                             vertexDirectionStrideInBytes;
615
617     OptixDisplacementMicromapBiasAndScaleFormat vertexBiasAndScaleFormat;
619     unsigned int                                vertexBiasAndScaleStrideInBytes;
620
622     unsigned int triangleFlagsStrideInBytes;
623
625     unsigned int                               numDisplacementMicromapUsageCounts;
628     const OptixDisplacementMicromapUsageCount* displacementMicromapUsageCounts;
629
630 } OptixBuildInputDisplacementMicromap;
631
632
636 typedef struct OptixBuildInputTriangleArray
637 {
645     const CUdeviceptr* vertexBuffers;
646
648     unsigned int numVertices;
```

```
649
651     OptixVertexFormat vertexFormat;
652
655     unsigned int vertexStrideInBytes;
656
660     CUdeviceptr indexBuffer;
661
663     unsigned int numIndexTriplets;
664
666     OptixIndicesFormat indexFormat;
667
670     unsigned int indexStrideInBytes;
671
675     CUdeviceptr preTransform;
676
680     const unsigned int* flags;
681
683     unsigned int numSbtRecords;
684
688     CUdeviceptr sbtIndexOffsetBuffer;
689
691     unsigned int sbtIndexOffsetSizeInBytes;
692
695     unsigned int sbtIndexOffsetStrideInBytes;
696
699     unsigned int primitiveIndexOffset;
700
702     OptixTransformFormat transformFormat;
703
705     OptixBuildInputOpacityMicromap opacityMicromap;
707     OptixBuildInputDisplacementMicromap displacementMicromap;
708
709 } OptixBuildInputTriangleArray;
710
714 typedef struct OptixRelocateInputTriangleArray
715 {
718     unsigned int numSbtRecords;
719
721     OptixRelocateInputOpacityMicromap opacityMicromap;
722 } OptixRelocateInputTriangleArray;
723
726 typedef enum OptixPrimitiveType
727 {
729     OPTIX_PRIMITIVE_TYPE_CUSTOM                         = 0x2500,
731     OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE        = 0x2501,
733     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE            = 0x2502,
735     OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR                   = 0x2503,
737     OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM               = 0x2504,
739     OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE         = 0x2505,
741     OPTIX_PRIMITIVE_TYPE_SPHERE                         = 0x2506,
743     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER             = 0x2507,
745     OPTIX_PRIMITIVE_TYPE_TRIANGLE                       = 0x2531,
747     OPTIX_PRIMITIVE_TYPE_DISPLACED_MICROMESH_TRIANGLE   = 0x2532,
748 } OptixPrimitiveType;
749
753 typedef enum OptixPrimitiveTypeFlags
754 {
756     OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM                       = 1 << 0,
758     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE      = 1 << 1,
760     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE          = 1 << 2,
762     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR                 = 1 << 3,
764     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM             = 1 << 4,
766     OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE       = 1 << 5,
768     OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE                       = 1 << 6,
770     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER           = 1 << 7,
772     OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE                     = 1 << 31,
774     OPTIX_PRIMITIVE_TYPE_FLAGS_DISPLACED_MICROMESH_TRIANGLE = 1 << 30,
```

```
775 } OptixPrimitiveTypeFlags;
776
779 typedef enum OptixCurveEndcapFlags
780 {
782     OPTIX_CURVE_ENDCAP_DEFAULT                          = 0,
784     OPTIX_CURVE_ENDCAP_ON                               = 1 « 0,
785 } OptixCurveEndcapFlags;
786
804 typedef struct OptixBuildInputCurveArray
805 {
808     OptixPrimitiveType curveType;
810     unsigned int numPrimitives;
811
816     const CUdeviceptr* vertexBuffers;
818     unsigned int numVertices;
821     unsigned int vertexStrideInBytes;
822
825     const CUdeviceptr* widthBuffers;
828     unsigned int widthStrideInBytes;
829
831     const CUdeviceptr* normalBuffers;
833     unsigned int normalStrideInBytes;
834
840     CUdeviceptr indexBuffer;
843     unsigned int indexStrideInBytes;
844
847     unsigned int flag;
848
851     unsigned int primitiveIndexOffset;
852
854     unsigned int endcapFlags;
855 } OptixBuildInputCurveArray;
856
869 typedef struct OptixBuildInputSphereArray
870 {
875   const CUdeviceptr* vertexBuffers;
876
879   unsigned int vertexStrideInBytes;
881   unsigned int numVertices;
882
885   const CUdeviceptr* radiusBuffers;
888   unsigned int radiusStrideInBytes;
891   int singleRadius;
892
896   const unsigned int* flags;
897
899   unsigned int numSbtRecords;
903   CUdeviceptr sbtIndexOffsetBuffer;
905   unsigned int sbtIndexOffsetSizeInBytes;
908   unsigned int sbtIndexOffsetStrideInBytes;
909
912   unsigned int primitiveIndexOffset;
913 } OptixBuildInputSphereArray;
914
916 typedef struct OptixAabb
917 {
918     float minX;
919     float minY;
920     float minZ;
921     float maxX;
922     float maxY;
923     float maxZ;
924 } OptixAabb;
925
929 typedef struct OptixBuildInputCustomPrimitiveArray
930 {
935     const CUdeviceptr* aabbBuffers;
```

```
936
939     unsigned int numPrimitives;
940
944     unsigned int strideInBytes;
945
949     const unsigned int* flags;
950
952     unsigned int numSbtRecords;
953
957     CUdeviceptr sbtIndexOffsetBuffer;
958
960     unsigned int sbtIndexOffsetSizeInBytes;
961
964     unsigned int sbtIndexOffsetStrideInBytes;
965
968     unsigned int primitiveIndexOffset;
969 } OptixBuildInputCustomPrimitiveArray;
970
974 typedef struct OptixBuildInputInstanceArray
975 {
983     CUdeviceptr instances;
984
986     unsigned int numInstances;
987
991     unsigned int instanceStride;
992 } OptixBuildInputInstanceArray;
993
997 typedef struct OptixRelocateInputInstanceArray
998 {
1001     unsigned int numInstances;
1002
1008     CUdeviceptr traversableHandles;
1009
1010 } OptixRelocateInputInstanceArray;
1011
1015 typedef enum OptixBuildInputType
1016 {
1018     OPTIX_BUILD_INPUT_TYPE_TRIANGLES = 0x2141,
1020     OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES = 0x2142,
1022     OPTIX_BUILD_INPUT_TYPE_INSTANCES = 0x2143,
1024     OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS = 0x2144,
1026     OPTIX_BUILD_INPUT_TYPE_CURVES = 0x2145,
1028     OPTIX_BUILD_INPUT_TYPE_SPHERES = 0x2146
1029 } OptixBuildInputType;
1030
1036 typedef struct OptixBuildInput
1037 {
1039     OptixBuildInputType type;
1040
1041     union
1042     {
1044         OptixBuildInputTriangleArray triangleArray;
1046         OptixBuildInputCurveArray curveArray;
1048         OptixBuildInputSphereArray sphereArray;
1050         OptixBuildInputCustomPrimitiveArray customPrimitiveArray;
1052         OptixBuildInputInstanceArray instanceArray;
1053         char pad[1024];
1054     };
1055 } OptixBuildInput;
1056
1060 typedef struct OptixRelocateInput
1061 {
1063     OptixBuildInputType type;
1064
1065     union
1066     {
1068         OptixRelocateInputInstanceArray instanceArray;
```

```
1069
1071         OptixRelocateInputTriangleArray triangleArray;
1072
1074     };
1075 } OptixRelocateInput;
1076
1077 // Some 32-bit tools use this header. This static_assert fails for them because
1078 // the default enum size is 4 bytes, rather than 8, under 32-bit compilers.
1079 // This #ifndef allows them to disable the static assert.
1080
1081 // TODO Define a static assert for C/pre-C++-11
1082 #if defined(__cplusplus) && __cplusplus >= 201103L
1083 static_assert(sizeof(OptixBuildInput) == 8 + 1024, "OptixBuildInput has wrong size");
1084 #endif
1085
1089 typedef enum OptixInstanceFlags
1090 {
1092     OPTIX_INSTANCE_FLAG_NONE = 0,
1093
1097     OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u « 0,
1098
1101     OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u « 1,
1102
1106     OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT = 1u « 2,
1107
1112     OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u « 3,
1113
1114
1116     OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u « 4,
1119     OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS = 1u « 5,
1120
1121 } OptixInstanceFlags;
1122
1126 typedef struct OptixInstance
1127 {
1129     float transform[12];
1130
1132     unsigned int instanceId;
1133
1137     unsigned int sbtOffset;
1138
1141     unsigned int visibilityMask;
1142
1144     unsigned int flags;
1145
1147     OptixTraversableHandle traversableHandle;
1148
1150     unsigned int pad[2];
1151 } OptixInstance;
1152
1156 typedef enum OptixBuildFlags
1157 {
1159     OPTIX_BUILD_FLAG_NONE = 0,
1160
1163     OPTIX_BUILD_FLAG_ALLOW_UPDATE = 1u « 0,
1164
1165     OPTIX_BUILD_FLAG_ALLOW_COMPACTION = 1u « 1,
1166
1168     OPTIX_BUILD_FLAG_PREFER_FAST_TRACE = 1u « 2,
1169
1171     OPTIX_BUILD_FLAG_PREFER_FAST_BUILD = 1u « 3,
1172
1182     OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS = 1u « 4,
1183
1186     OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS = 1u « 5,
1187
1191     OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE = 1u « 6,
```

```
1192
1196     OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1u « 7,
1197 } OptixBuildFlags;
1198
1199
1201 typedef enum OptixOpacityMicromapFlags
1202 {
1203     OPTIX_OPACITY_MICROMAP_FLAG_NONE = 0,
1204
1206     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 « 0,
1207
1209     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 « 1,
1210 } OptixOpacityMicromapFlags;
1211
1213 typedef struct OptixOpacityMicromapDesc
1214 {
1216     unsigned int  byteOffset;
1218     unsigned short subdivisionLevel;
1220     unsigned short format;
1221 } OptixOpacityMicromapDesc;
1222
1227 typedef struct OptixOpacityMicromapHistogramEntry
1228 {
1230     unsigned int               count;
1232     unsigned int               subdivisionLevel;
1234     OptixOpacityMicromapFormat format;
1235 } OptixOpacityMicromapHistogramEntry;
1236
1238 typedef struct OptixOpacityMicromapArrayBuildInput
1239 {
1241     unsigned int flags;
1242
1244     CUdeviceptr inputBuffer;
1245
1248     CUdeviceptr perMicromapDescBuffer;
1249
1253     unsigned int perMicromapDescStrideInBytes;
1254
1256     unsigned int numMicromapHistogramEntries;
1259     const OptixOpacityMicromapHistogramEntry* micromapHistogramEntries;
1260 } OptixOpacityMicromapArrayBuildInput;
1261
1263 typedef struct OptixMicromapBufferSizes
1264 {
1265     size_t outputSizeInBytes;
1266     size_t tempSizeInBytes;
1267 } OptixMicromapBufferSizes;
1268
1270 typedef struct OptixMicromapBuffers
1271 {
1273     CUdeviceptr output;
1275     size_t outputSizeInBytes;
1277     CUdeviceptr temp;
1279     size_t tempSizeInBytes;
1280 } OptixMicromapBuffers;
1281
1282
1294 typedef enum OptixBuildOperation
1295 {
1297     OPTIX_BUILD_OPERATION_BUILD = 0x2161,
1299     OPTIX_BUILD_OPERATION_UPDATE = 0x2162,
1300 } OptixBuildOperation;
1301
1305 typedef enum OptixMotionFlags
1306 {
1307     OPTIX_MOTION_FLAG_NONE         = 0,
1308     OPTIX_MOTION_FLAG_START_VANISH = 1u « 0,
```

```
1309     OPTIX_MOTION_FLAG_END_VANISH   = 1u « 1
1310 } OptixMotionFlags;
1311
1316 typedef struct OptixMotionOptions
1317 {
1320     unsigned short numKeys;
1321
1323     unsigned short flags;
1324
1326     float timeBegin;
1327
1329     float timeEnd;
1330 } OptixMotionOptions;
1331
1335 typedef struct OptixAccelBuildOptions
1336 {
1338     unsigned int buildFlags;
1339
1346     OptixBuildOperation operation;
1347
1349     OptixMotionOptions motionOptions;
1350 } OptixAccelBuildOptions;
1351
1357 typedef struct OptixAccelBufferSizes
1358 {
1361     size_t outputSizeInBytes;
1362
1365     size_t tempSizeInBytes;
1366
1371     size_t tempUpdateSizeInBytes;
1372 } OptixAccelBufferSizes;
1373
1377 typedef enum OptixAccelPropertyType
1378 {
1380     OPTIX_PROPERTY_TYPE_COMPACTED_SIZE = 0x2181,
1381
1383     OPTIX_PROPERTY_TYPE_AABBS = 0x2182,
1384 } OptixAccelPropertyType;
1385
1389 typedef struct OptixAccelEmitDesc
1390 {
1392     CUdeviceptr result;
1393
1395     OptixAccelPropertyType type;
1396 } OptixAccelEmitDesc;
1397
1402 typedef struct OptixRelocationInfo
1403 {
1405     unsigned long long info[4];
1406 } OptixRelocationInfo;
1407
1413 typedef struct OptixStaticTransform
1414 {
1416     OptixTraversableHandle child;
1417
1419     unsigned int pad[2];
1420
1422     float transform[12];
1423
1426     float invTransform[12];
1427 } OptixStaticTransform;
1428
1453 typedef struct OptixMatrixMotionTransform
1454 {
1456     OptixTraversableHandle child;
1457
1460     OptixMotionOptions motionOptions;
```

```
1461
1463     unsigned int pad[3];
1464
1466     float transform[2][12];
1467 } OptixMatrixMotionTransform;
1468
1476 //      [ sx   a   b  pvx ]
1477 //  S = [  0  sy   c  pvy ]
1478 //      [  0   0  sz  pvz ]
1487 //      [  1  0  0 tx ]
1488 //  T = [  0  1  0 ty ]
1489 //      [  0  0  1 tz ]
1499 typedef struct OptixSRTData
1500 {
1503     float sx, a, b, pvx, sy, c, pvy, sz, pvz, qx, qy, qz, qw, tx, ty, tz;
1505 } OptixSRTData;
1506
1507 // TODO Define a static assert for C/pre-C++-11
1508 #if defined(__cplusplus) && __cplusplus >= 201103L
1509 static_assert(sizeof(OptixSRTData) == 16 * 4, "OptixSRTData has wrong size");
1510 #endif
1511
1536 typedef struct OptixSRTMotionTransform
1537 {
1539     OptixTraversableHandle child;
1540
1543     OptixMotionOptions motionOptions;
1544
1546     unsigned int pad[3];
1547
1549     OptixSRTData srtData[2];
1550 } OptixSRTMotionTransform;
1551
1552 // TODO Define a static assert for C/pre-C++-11
1553 #if defined(__cplusplus) && __cplusplus >= 201103L
1554 static_assert(sizeof(OptixSRTMotionTransform) == 8 + 12 + 12 + 2 * 16 * 4, "OptixSRTMotionTransform has
wrong size");
1555 #endif
1556
1560 typedef enum OptixTraversableType
1561 {
1563     OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM = 0x21C1,
1565     OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM = 0x21C2,
1567     OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM = 0x21C3,
1568 } OptixTraversableType;
1569
1573 typedef enum OptixPixelFormat
1574 {
1575     OPTIX_PIXEL_FORMAT_HALF1  = 0x220a,
1576     OPTIX_PIXEL_FORMAT_HALF2  = 0x2207,
1577     OPTIX_PIXEL_FORMAT_HALF3  = 0x2201,
1578     OPTIX_PIXEL_FORMAT_HALF4  = 0x2202,
1579     OPTIX_PIXEL_FORMAT_FLOAT1 = 0x220b,
1580     OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208,
1581     OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203,
1582     OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204,
1583     OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205,
1584     OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206,
1585     OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209
1586 } OptixPixelFormat;
1587
1591 typedef struct OptixImage2D
1592 {
1594     CUdeviceptr data;
1596     unsigned int width;
1598     unsigned int height;
1600     unsigned int rowStrideInBytes;
```

```
1605     unsigned int pixelStrideInBytes;
1607     OptixPixelFormat format;
1608 } OptixImage2D;
1609
1613 typedef enum OptixDenoiserModelKind
1614 {
1616     OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322,
1617
1619     OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323,
1620
1622     OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324,
1623
1625     OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325,
1626
1628     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326,
1629
1631     OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327,
1632
1635     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328
1636 } OptixDenoiserModelKind;
1637
1641 typedef enum OptixDenoiserAlphaMode
1642 {
1644     OPTIX_DENOISER_ALPHA_MODE_COPY = 0,
1645
1647     OPTIX_DENOISER_ALPHA_MODE_DENOISE = 1
1648 } OptixDenoiserAlphaMode;
1649
1653 typedef struct OptixDenoiserOptions
1654 {
1655     // if nonzero, albedo image must be given in OptixDenoiserGuideLayer
1656     unsigned int guideAlbedo;
1657
1658     // if nonzero, normal image must be given in OptixDenoiserGuideLayer
1659     unsigned int guideNormal;
1660
1662     OptixDenoiserAlphaMode denoiseAlpha;
1663 } OptixDenoiserOptions;
1664
1668 typedef struct OptixDenoiserGuideLayer
1669 {
1670     // image with three components: R, G, B.
1671     OptixImage2D  albedo;
1672
1673     // image with two or three components: X, Y, Z.
1674     // (X, Y) camera space. (X, Y, Z) world space, depending on model.
1675     OptixImage2D  normal;
1676
1677     // image with two components: X, Y.
1678     // pixel movement from previous to current frame for each pixel in screen space.
1679     OptixImage2D  flow;
1680
1681     // Internal images used in temporal AOV denoising modes,
1682     // pixel format OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER.
1683     OptixImage2D  previousOutputInternalGuideLayer;
1684     OptixImage2D  outputInternalGuideLayer;
1685
1686     // image with a single component value that specifies how trustworthy the flow vector at x,y
position in
1687     // OptixDenoiserGuideLayer::flow is. Range 0..1 (low->high trustworthiness).
1688     // Ignored if data pointer in the image is zero.
1689     OptixImage2D flowTrustworthiness;
1690
1691 } OptixDenoiserGuideLayer;
1692
1695 typedef enum OptixDenoiserAOVType
1696 {
```

```
1698     OPTIX_DENOISER_AOV_TYPE_NONE       = 0,
1699
1700     OPTIX_DENOISER_AOV_TYPE_BEAUTY     = 0x7000,
1701     OPTIX_DENOISER_AOV_TYPE_SPECULAR   = 0x7001,
1702     OPTIX_DENOISER_AOV_TYPE_REFLECTION = 0x7002,
1703     OPTIX_DENOISER_AOV_TYPE_REFRACTION = 0x7003,
1704     OPTIX_DENOISER_AOV_TYPE_DIFFUSE    = 0x7004
1705
1706 } OptixDenoiserAOVType;
1707
1711 typedef struct OptixDenoiserLayer
1712 {
1713     // input image (beauty or AOV)
1714     OptixImage2D  input;
1715
1716     // denoised output image from previous frame if temporal model kind selected
1717     OptixImage2D  previousOutput;
1718
1719     // denoised output for given input
1720     OptixImage2D  output;
1721
1722     // Type of AOV, used in temporal AOV modes as a hint to improve image quality.
1723     OptixDenoiserAOVType type;
1724 } OptixDenoiserLayer;
1725
1731
1732 typedef struct OptixDenoiserParams
1733 {
1738     CUdeviceptr  hdrIntensity;
1739
1744     float        blendFactor;
1745
1751     CUdeviceptr  hdrAverageColor;
1752
1757     unsigned int temporalModeUsePreviousLayers;
1758 } OptixDenoiserParams;
1759
1763 typedef struct OptixDenoiserSizes
1764 {
1766     size_t stateSizeInBytes;
1767
1770     size_t withOverlapScratchSizeInBytes;
1771
1774     size_t withoutOverlapScratchSizeInBytes;
1775
1777     unsigned int overlapWindowSizeInPixels;
1778
1781     size_t computeAverageColorSizeInBytes;
1782
1785     size_t computeIntensitySizeInBytes;
1786
1788     size_t internalGuideLayerPixelSizeInBytes;
1789 } OptixDenoiserSizes;
1790
1795 typedef enum OptixRayFlags
1796 {
1798     OPTIX_RAY_FLAG_NONE = 0u,
1799
1804     OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u « 0,
1805
1810     OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u « 1,
1811
1814     OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u « 2,
1815
1817     OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u « 3,
1818
1823     OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u « 4,
```

```
1824
1829     OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u « 5,
1830
1836     OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u « 6,
1837
1843     OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u « 7,
1844
1846     OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u « 10,
1847 } OptixRayFlags;
1848
1854 typedef enum OptixTransformType
1855 {
1856     OPTIX_TRANSFORM_TYPE_NONE                     = 0,
1857     OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM         = 1,
1858     OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM  = 2,
1859     OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM     = 3,
1860     OPTIX_TRANSFORM_TYPE_INSTANCE                 = 4,
1861 } OptixTransformType;
1862
1865 typedef enum OptixTraversableGraphFlags
1866 {
1869     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0,
1870
1874     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u « 0,
1875
1880     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u « 1,
1881 } OptixTraversableGraphFlags;
1882
1886 typedef enum OptixCompileOptimizationLevel
1887 {
1889     OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0,
1891     OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340,
1893     OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341,
1895     OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342,
1897     OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343,
1898 } OptixCompileOptimizationLevel;
1899
1903 typedef enum OptixCompileDebugLevel
1904 {
1906     OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT  = 0,
1908     OPTIX_COMPILE_DEBUG_LEVEL_NONE     = 0x2350,
1911     OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL  = 0x2351,
1913     OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353,
1915     OPTIX_COMPILE_DEBUG_LEVEL_FULL     = 0x2352,
1916 } OptixCompileDebugLevel;
1917
1921 typedef enum OptixModuleCompileState
1922 {
1924     OPTIX_MODULE_COMPILE_STATE_NOT_STARTED       = 0x2360,
1925
1927     OPTIX_MODULE_COMPILE_STATE_STARTED           = 0x2361,
1928
1930     OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE = 0x2362,
1931
1933     OPTIX_MODULE_COMPILE_STATE_FAILED            = 0x2363,
1934
1936     OPTIX_MODULE_COMPILE_STATE_COMPLETED         = 0x2364,
1937 } OptixModuleCompileState;
1938
1939
1940
1973 typedef struct OptixModuleCompileBoundValueEntry {
1974     size_t pipelineParamOffsetInBytes;
1975     size_t sizeInBytes;
1976     const void* boundValuePtr;
1977     const char* annotation; // optional string to display, set to 0 if unused.  If unused,
1978                             // OptiX will report the annotation as "No annotation"
```

```
1979 } OptixModuleCompileBoundValueEntry;
1980
1982 typedef enum OptixPayloadTypeID {
1983     OPTIX_PAYLOAD_TYPE_DEFAULT = 0,
1984     OPTIX_PAYLOAD_TYPE_ID_0 = (1 « 0u),
1985     OPTIX_PAYLOAD_TYPE_ID_1 = (1 « 1u),
1986     OPTIX_PAYLOAD_TYPE_ID_2 = (1 « 2u),
1987     OPTIX_PAYLOAD_TYPE_ID_3 = (1 « 3u),
1988     OPTIX_PAYLOAD_TYPE_ID_4 = (1 « 4u),
1989     OPTIX_PAYLOAD_TYPE_ID_5 = (1 « 5u),
1990     OPTIX_PAYLOAD_TYPE_ID_6 = (1 « 6u),
1991     OPTIX_PAYLOAD_TYPE_ID_7 = (1 « 7u)
1992 } OptixPayloadTypeID;
1993
2007 typedef enum OptixPayloadSemantics
2008 {
2009     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE       = 0,
2010     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ       = 1u « 0,
2011     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE      = 2u « 0,
2012     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u « 0,
2013
2014     OPTIX_PAYLOAD_SEMANTICS_CH_NONE                 = 0,
2015     OPTIX_PAYLOAD_SEMANTICS_CH_READ                 = 1u « 2,
2016     OPTIX_PAYLOAD_SEMANTICS_CH_WRITE                = 2u « 2,
2017     OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE           = 3u « 2,
2018
2019     OPTIX_PAYLOAD_SEMANTICS_MS_NONE                 = 0,
2020     OPTIX_PAYLOAD_SEMANTICS_MS_READ                 = 1u « 4,
2021     OPTIX_PAYLOAD_SEMANTICS_MS_WRITE                = 2u « 4,
2022     OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE           = 3u « 4,
2023
2024     OPTIX_PAYLOAD_SEMANTICS_AH_NONE                 = 0,
2025     OPTIX_PAYLOAD_SEMANTICS_AH_READ                 = 1u « 6,
2026     OPTIX_PAYLOAD_SEMANTICS_AH_WRITE                = 2u « 6,
2027     OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE           = 3u « 6,
2028
2029     OPTIX_PAYLOAD_SEMANTICS_IS_NONE                 = 0,
2030     OPTIX_PAYLOAD_SEMANTICS_IS_READ                 = 1u « 8,
2031     OPTIX_PAYLOAD_SEMANTICS_IS_WRITE                = 2u « 8,
2032     OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE           = 3u « 8,
2033 } OptixPayloadSemantics;
2034
2036 typedef struct OptixPayloadType
2037 {
2039     unsigned int numPayloadValues;
2040
2042     const unsigned int *payloadSemantics;
2043 } OptixPayloadType;
2044
2048 typedef struct OptixModuleCompileOptions
2049 {
2052     int maxRegisterCount;
2053
2055     OptixCompileOptimizationLevel optLevel;
2056
2058     OptixCompileDebugLevel debugLevel;
2059
2061     const OptixModuleCompileBoundValueEntry* boundValues;
2062
2064     unsigned int numBoundValues;
2065
2068     unsigned int numPayloadTypes;
2069
2071     const OptixPayloadType* payloadTypes;
2072
2073 } OptixModuleCompileOptions;
2074
```

```
2076 typedef enum OptixProgramGroupKind
2077 {
2080     OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421,
2081
2084     OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422,
2085
2088     OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423,
2089
2092     OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424,
2093
2096     OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425
2097 } OptixProgramGroupKind;
2098
2100 typedef enum OptixProgramGroupFlags
2101 {
2103     OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0
2104 } OptixProgramGroupFlags;
2105
2112 typedef struct OptixProgramGroupSingleModule
2113 {
2115     OptixModule module;
2117     const char* entryFunctionName;
2118 } OptixProgramGroupSingleModule;
2119
2125 typedef struct OptixProgramGroupHitgroup
2126 {
2128     OptixModule moduleCH;
2130     const char* entryFunctionNameCH;
2132     OptixModule moduleAH;
2134     const char* entryFunctionNameAH;
2136     OptixModule moduleIS;
2138     const char* entryFunctionNameIS;
2139 } OptixProgramGroupHitgroup;
2140
2146 typedef struct OptixProgramGroupCallables
2147 {
2149     OptixModule moduleDC;
2151     const char* entryFunctionNameDC;
2153     OptixModule moduleCC;
2155     const char* entryFunctionNameCC;
2156 } OptixProgramGroupCallables;
2157
2159 typedef struct OptixProgramGroupDesc
2160 {
2162     OptixProgramGroupKind kind;
2163
2165     unsigned int flags;
2166
2167     union
2168     {
2170         OptixProgramGroupSingleModule raygen;
2172         OptixProgramGroupSingleModule miss;
2174         OptixProgramGroupSingleModule exception;
2176         OptixProgramGroupCallables callables;
2178         OptixProgramGroupHitgroup hitgroup;
2179     };
2180 } OptixProgramGroupDesc;
2181
2185 typedef struct OptixProgramGroupOptions
2186 {
2199     const OptixPayloadType* payloadType;
2200 } OptixProgramGroupOptions;
2201
2203 typedef enum OptixExceptionCodes
2204 {
2207     OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1,
2208
```

```
2211     OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2,
2212
2213
2214 } OptixExceptionCodes;
2215
2219 typedef enum OptixExceptionFlags
2220 {
2222     OPTIX_EXCEPTION_FLAG_NONE = 0,
2223
2230     OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u « 0,
2231
2238     OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u « 1,
2239
2242     OPTIX_EXCEPTION_FLAG_USER = 1u « 2,
2243
2244 } OptixExceptionFlags;
2245
2251 typedef struct OptixPipelineCompileOptions
2252 {
2254     int usesMotionBlur;
2255
2257     unsigned int traversableGraphFlags;
2258
2261     int numPayloadValues;
2262
2265     int numAttributeValues;
2266
2268     unsigned int exceptionFlags;
2269
2273     const char* pipelineLaunchParamsVariableName;
2274
2277     unsigned int usesPrimitiveTypeFlags;
2278
2280     int allowOpacityMicromaps;
2281 } OptixPipelineCompileOptions;
2282
2286 typedef struct OptixPipelineLinkOptions
2287 {
2290     unsigned int maxTraceDepth;
2291
2292 } OptixPipelineLinkOptions;
2293
2297 typedef struct OptixShaderBindingTable
2298 {
2301     CUdeviceptr raygenRecord;
2302
2305     CUdeviceptr exceptionRecord;
2306
2310     CUdeviceptr  missRecordBase;
2311     unsigned int missRecordStrideInBytes;
2312     unsigned int missRecordCount;
2314
2318     CUdeviceptr  hitgroupRecordBase;
2319     unsigned int hitgroupRecordStrideInBytes;
2320     unsigned int hitgroupRecordCount;
2322
2327     CUdeviceptr  callablesRecordBase;
2328     unsigned int callablesRecordStrideInBytes;
2329     unsigned int callablesRecordCount;
2331
2332 } OptixShaderBindingTable;
2333
2337 typedef struct OptixStackSizes
2338 {
2340     unsigned int cssRG;
2342     unsigned int cssMS;
2344     unsigned int cssCH;
```

```
2346     unsigned int cssAH;
2348     unsigned int cssIS;
2350     unsigned int cssCC;
2352     unsigned int dssDC;
2353
2354 } OptixStackSizes;
2355
2357 typedef enum OptixQueryFunctionTableOptions
2358 {
2360     OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0
2361
2362 } OptixQueryFunctionTableOptions;
2363
2365 typedef OptixResult(OptixQueryFunctionTable_t)(int          abiId,
2366                                                unsigned int numOptions,
2367                                                OptixQueryFunctionTableOptions* /*optionKeys*/,
2368                                                const void** /*optionValues*/,
2369                                                void*  functionTable,
2370                                                size_t sizeOfTable);
2371
2376 typedef struct OptixBuiltinISOptions
2377 {
2378     OptixPrimitiveType        builtinISModuleType;
2380     int                       usesMotionBlur;
2382     unsigned int              buildFlags;
2384     unsigned int              curveEndcapFlags;
2385 } OptixBuiltinISOptions;
2386
2387 #if defined(__CUDACC__)
2392 typedef struct OptixInvalidRayExceptionDetails
2393 {
2394     float3 origin;
2395     float3 direction;
2396     float  tmin;
2397     float  tmax;
2398     float  time;
2399 } OptixInvalidRayExceptionDetails;
2400
2407 typedef struct OptixParameterMismatchExceptionDetails
2408 {
2410     unsigned int expectedParameterCount;
2412     unsigned int passedArgumentCount;
2414     unsigned int sbtIndex;
2416     char*        callableName;
2417 } OptixParameterMismatchExceptionDetails;
2418 #endif
2419
2420   // end group optix_types
2422
2423 #endif  // OPTIX_OPTIX_TYPES_H
```

8.27 main.dox File Reference