NVIDIA

NVIDIA OptiX 9.0

API Reference Manual

30 January 2025
Version 9.0

Table of Contents

1 NVIDIA OptiX 9.0 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

Modules

- Cooperative Vector

Classes

- struct OptixIncomingHitObject
- struct OptixOutgoingHitObject

Functions

- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTraverse (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTraverse (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- static __forceinline__ __device__ void optixReorder (unsigned int coherenceHint, unsigned int numCoherenceHintBitsFromLSB)
- static __forceinline__ __device__ void optixReorder ()
- template<typename... Payload>
  static __forceinline__ __device__ void optixInvoke (Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixInvoke (OptixPayloadTypeID type, Payload &... payload)
- static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float rayTime, unsigned int rayFlags, OptixTraverseData traverseData, const OptixTraversableHandle *transforms, unsigned int numTransforms)
- static __forceinline__ __device__ void optixMakeMissHitObject (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int rayFlags)
- static __forceinline__ __device__ void optixMakeNopHitObject ()
- static __forceinline__ __device__ void optixHitObjectGetTraverseData (OptixTraverseData *data)
- static __forceinline__ __device__ bool optixHitObjectIsHit ()
- static __forceinline__ __device__ bool optixHitObjectIsMiss ()
- static __forceinline__ __device__ bool optixHitObjectIsNop ()

- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex ()
- static __forceinline__ __device__ void optixHitObjectSetSbtRecordIndex (unsigned int sbtRecordIndex)
- static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetGASTraversableHandle ()
- static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_5 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_6 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_7 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_8 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_9 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_10 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_11 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_12 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_13 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_14 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_15 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_16 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_17 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_18 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_19 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_20 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_21 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_22 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_23 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_24 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_25 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_26 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_27 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_28 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_29 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_30 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_31 (unsigned int p)
- static __forceinline__ __device__ unsigned int optixGetPayload_0 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_1 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_2 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_3 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_4 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_5 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_6 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_7 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_8 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_9 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_10 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_11 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_12 ()

- static __forceinline__ __device__ unsigned int optixGetPayload_13 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_14 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_15 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_16 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_17 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_18 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_19 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_20 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_21 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_22 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_23 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_24 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_25 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_26 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_27 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_28 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_29 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_30 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_31 ()
- static __forceinline__ __device__ void optixSetPayloadTypes (unsigned int typeMask)
- static __forceinline__ __device__ unsigned int optixUndefinedValue ()
- static __forceinline__ __device__ float3 optixGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixGetObjectRayOrigin ()
- static __forceinline__ __device__ float3 optixGetObjectRayDirection ()
- static __forceinline__ __device__ float optixGetRayTmin ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmin ()
- static __forceinline__ __device__ float optixGetRayTmax ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmax ()
- static __forceinline__ __device__ float optixGetRayTime ()
- static __forceinline__ __device__ float optixHitObjectGetRayTime ()
- static __forceinline__ __device__ unsigned int optixGetRayFlags ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetRayFlags ()
- static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceTraversableFromIAS (OptixTraversableHandle ias, unsigned int instIdx)
- static __forceinline__ __device__ void optixGetTriangleVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static __forceinline__ __device__ void optixGetTriangleVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static __forceinline__ __device__ void optixGetTriangleVertexData (float3 data[3])
- static __forceinline__ __device__ void optixHitObjectGetTriangleVertexData (float3 data[3])
- static __forceinline__ __device__ void optixGetLinearCurveVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void optixGetLinearCurveVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])

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

- static __forceinline__ __device__ void optixGetCubicBezierVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCubicBezierVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCubicBezierRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetRibbonVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetRibbonVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetRibbonVertexData (float4 data[3])
- static __forceinline__ __device__ void optixHitObjectGetRibbonVertexData (float4 data[3])
- static __forceinline__ __device__ float3 optixGetRibbonNormal (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ float3 optixGetRibbonNormalFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ float3 optixGetRibbonNormal (float2 ribbonParameters)
- static __forceinline__ __device__ float3 optixHitObjectGetRibbonNormal (float2 ribbonParameters)
- static __forceinline__ __device__ void optixGetSphereData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ void optixGetSphereDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ void optixGetSphereData (float4 data[1])
- static __forceinline__ __device__ void optixHitObjectGetSphereData (float4 data[1])
- static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle ()
- static __forceinline__ __device__ float optixGetGASMotionTimeBegin (OptixTraversableHandle gas)
- static __forceinline__ __device__ float optixGetGASMotionTimeEnd (OptixTraversableHandle gas)
- static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (OptixTraversableHandle gas)
- static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (float m[12])
- static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (float3 normal)
- static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (float3 normal)
- static __forceinline__ __device__ void optixHitObjectGetWorldToObjectTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixHitObjectGetObjectToWorldTransformMatrix (float m[12])

- static __forceinline__ __device__ float3 optixHitObjectTransformPointFromWorldToObjectSpace (float3 point)
- static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromWorldToObjectSpace (float3 vec)
- static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromWorldToObjectSpace (float3 normal)
- static __forceinline__ __device__ float3 optixHitObjectTransformPointFromObjectToWorldSpace (float3 point)
- static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromObjectToWorldSpace (float3 vec)
- static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromObjectToWorldSpace (float3 normal)
- template<typename HitState >
  static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (const HitState &hs, float m[12])
- template<typename HitState >
  static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (const HitState &hs, float m[12])
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (const HitState &hs, float3 point)
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (const HitState &hs, float3 vec)
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (const HitState &hs, float3 normal)
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (const HitState &hs, float3 point)
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (const HitState &hs, float3 vec)
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (const HitState &hs, float3 normal)
- static __forceinline__ __device__ unsigned int optixGetTransformListSize ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTransformType optixGetTransformTypeFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixStaticTransform * optixGetStaticTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixSRTMotionTransform * optixGetSRTMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixMatrixMotionTransform * optixGetMatrixMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle (OptixTraversableHandle handle)

- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceChildFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceInverseTransformFromHandle (OptixTraversableHandle handle)
- static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6, unsigned int a7)
- static __forceinline__ __device__ unsigned int optixGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_7 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7 ()
- static __forceinline__ __device__ void optixTerminateRay ()
- static __forceinline__ __device__ void optixIgnoreIntersection ()
- static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixGetClusterId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetClusterId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ()

- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixGetHitKind ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind ()
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsFrontFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsBackFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ()
- static __forceinline__ __device__ bool optixIsFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsBackFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleHit ()
- static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ()
- static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ()
- static __forceinline__ __device__ float2 optixHitObjectGetTriangleBarycentrics ()
- static __forceinline__ __device__ float optixGetCurveParameter ()
- static __forceinline__ __device__ float optixHitObjectGetCurveParameter ()
- static __forceinline__ __device__ float2 optixGetRibbonParameters ()
- static __forceinline__ __device__ float2 optixHitObjectGetRibbonParameters ()
- static __forceinline__ __device__ uint3 optixGetLaunchIndex ()
- static __forceinline__ __device__ uint3 optixGetLaunchDimensions ()
- static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ()
- static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer ()
- static __forceinline__ __device__ void optixThrowException (int exceptionCode)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6, unsigned int exceptionDetail7)

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

5.1.1 Detailed Description

OptiX Device API.

5.1.2 Function Documentation

5.1.2.1 optixContinuationCall()

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

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

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

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

As opposed to direct callable programs, continuation callable programs are allowed to 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 optixGetCatmullRomRocapsVertexData()

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

5.1.2.12 optixGetCatmullRomRocapsVertexDataFromHandle()

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

5.1.2.13 optixGetCatmullRomVertexData() [1/2]

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

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

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

Available in AH, CH

5.1.2.14 optixGetCatmullRomVertexData() [2/2]

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

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

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

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

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

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

Available in all OptiX program types

5.1.2.15 optixGetCatmullRomVertexDataFromHandle()

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

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

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

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

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

Available in all OptiX program types

5.1.2.16 optixGetClusterId()

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

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

Returns OPTIX_CLUSTER_ID_INVALID if a non-Cluster GAS was intersected.

Available in AH, CH

5.1.2.17 optixGetCubicBezierRocapsVertexData()

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

5.1.2.18 optixGetCubicBezierRocapsVertexDataFromHandle()

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

5.1.2.19 optixGetCubicBezierVertexData() [1/2]

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

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

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

Available in AH, CH

5.1.2.20 optixGetCubicBezierVertexData() [2/2]

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

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

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

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

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

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

Available in all OptiX program types

5.1.2.21 optixGetCubicBezierVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetCubicBezierVertexDataFromHandle (
```

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

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

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

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

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

Available in all OptiX program types

5.1.2.22 optixGetCubicBSplineRocapsVertexData()

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

5.1.2.23 optixGetCubicBSplineRocapsVertexDataFromHandle()

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

5.1.2.24 optixGetCubicBSplineVertexData() [1/2]

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

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

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

Available in AH, CH

5.1.2.25 optixGetCubicBSplineVertexData() [2/2]

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

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

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

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

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

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

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

Available in all OptiX program types

5.1.2.26 optixGetCubicBSplineVertexDataFromHandle()

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

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

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

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

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

Available in all OptiX program types

5.1.2.27 optixGetCurveParameter()

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

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

Available in AH, CH

5.1.2.28 optixGetExceptionCode()

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

Returns the exception code.

Available in EX

5.1.2.29 optixGetExceptionDetail_0()

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

[static]

Returns the 32-bit exception detail at slot 0.

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

Available in EX

5.1.2.30 optixGetExceptionDetail_1( )

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

Returns the 32-bit exception detail at slot 1.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.31 optixGetExceptionDetail_2( )

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

Returns the 32-bit exception detail at slot 2.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.32 optixGetExceptionDetail_3( )

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

Returns the 32-bit exception detail at slot 3.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.33 optixGetExceptionDetail_4( )

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

Returns the 32-bit exception detail at slot 4.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.34 optixGetExceptionDetail_5( )

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

Returns the 32-bit exception detail at slot 5.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.35 optixGetExceptionDetail_6( )

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

Returns the 32-bit exception detail at slot 6.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.36 optixGetExceptionDetail_7()

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

Returns the 32-bit exception detail at slot 7.

See also optixGetExceptionDetail_0( ) Available in EX

5.1.2.37 optixGetExceptionInvalidRay()

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

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

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

Available in EX

5.1.2.38 optixGetExceptionInvalidSbtOffset()

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

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

Returns zero for all other exception codes.

Available in EX

5.1.2.39 optixGetExceptionInvalidTraversable()

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

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

Returns zero for all other exception codes.

Available in EX

5.1.2.40 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

5.1.2.41 optixGetExceptionParameterMismatch()

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

Returns information about an exception with code OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH.

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

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

Available in EX

5.1.2.42 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

5.1.2.43 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

5.1.2.44 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

5.1.2.45 optixGetGASPointerFromHandle()

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

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

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

Available in all OptiX program types

5.1.2.46 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

5.1.2.47 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

5.1.2.48 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

5.1.2.49 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

5.1.2.50 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

5.1.2.51 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

5.1.2.52 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

5.1.2.53 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

5.1.2.54 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

5.1.2.55 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

5.1.2.56 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

5.1.2.57 optixGetLinearCurveVertexData() [1/2]

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

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

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

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

Available in AH, CH

5.1.2.58 optixGetLinearCurveVertexData() [2/2]

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

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

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

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

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

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

Available in all OptiX program types

5.1.2.59 optixGetLinearCurveVertexDataFromHandle()

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

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

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

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

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

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

Available in all OptiX program types

5.1.2.60 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

5.1.2.61 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

5.1.2.62 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

5.1.2.63 optixGetObjectToWorldTransformMatrix() [1/2]

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

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

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

Available in IS, AH, CH

5.1.2.64 optixGetObjectToWorldTransformMatrix() [2/2]

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

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

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

Available in IS, AH, CH

5.1.2.65 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

5.1.2.66 optixGetPayload_1()

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

5.1.2.67 optixGetPayload_10()

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

5.1.2.68 optixGetPayload_11()

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

5.1.2.69 optixGetPayload_12()

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

5.1.2.70 optixGetPayload_13()

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

5.1.2.71 optixGetPayload_14()

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

5.1.2.72 optixGetPayload_15()

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

5.1.2.73 optixGetPayload_16()

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

5.1.2.74 optixGetPayload_17()

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

5.1.2.75 optixGetPayload_18()

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

5.1.2.76 optixGetPayload_19()

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

5.1.2.77 optixGetPayload_2()

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

5.1.2.78 optixGetPayload_20()

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

5.1.2.79 optixGetPayload_21()

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

5.1.2.80 optixGetPayload_22()

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

5.1.2.81 optixGetPayload_23()

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

5.1.2.82 optixGetPayload_24()

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

5.1.2.83 optixGetPayload_25()

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

5.1.2.84 optixGetPayload_26()

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

5.1.2.85 optixGetPayload_27()

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

5.1.2.86 optixGetPayload_28()

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

5.1.2.87 optixGetPayload_29()

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

5.1.2.88 optixGetPayload_3()

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

5.1.2.89 optixGetPayload_30()

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

5.1.2.90 optixGetPayload_31()

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

5.1.2.91 optixGetPayload_4()

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

5.1.2.92 optixGetPayload_5()

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

5.1.2.93 optixGetPayload_6()

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

5.1.2.94 optixGetPayload_7()

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

5.1.2.95 optixGetPayload_8()

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

5.1.2.96 optixGetPayload_9()

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

5.1.2.97 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

5.1.2.98 optixGetPrimitiveType() [1/2]

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

Function interpreting the hit kind associated with the current optixReportIntersection.

Available in AH, CH

5.1.2.99 optixGetPrimitiveType() [2/2]

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

Function interpreting the result of optixGetHitKind().

Available in all OptiX program types

5.1.2.100 optixGetQuadraticBSplineRocapsVertexData()

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

5.1.2.101 optixGetQuadraticBSplineRocapsVertexDataFromHandle()

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

5.1.2.102 optixGetQuadraticBSplineVertexData() [1/2]

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

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

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

Available in AH, CH

5.1.2.103 optixGetQuadraticBSplineVertexData() [2/2]

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

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

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

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

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

Available in all OptiX program types

5.1.2.104 optixGetQuadraticBSplineVertexDataFromHandle()

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

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

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

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

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

Available in all OptiX program types

5.1.2.105 optixGetRayFlags()

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

Returns the rayFlags passed into optixTrace.

Available in IS, AH, CH, MS

5.1.2.106 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

5.1.2.107 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

5.1.2.108 optixGetRayTmin()

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

Returns the tmin passed into optixTrace.

Available in IS, AH, CH, MS

5.1.2.109 optixGetRayVisibilityMask()

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

Returns the visibilityMask passed into optixTrace.

Available in IS, AH, CH, MS

5.1.2.110 optixGetRibbonNormal() [1/2]

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

Return ribbon normal at intersection reported by optixReportIntersection.

Available in AH, CH

5.1.2.111 optixGetRibbonNormal() [2/2]

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

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

Returns ribbon normal at intersection reported by optixReportIntersection.

Available in all OptiX program types

5.1.2.112 optixGetRibbonNormalFromHandle()

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

Returns ribbon normal at intersection reported by optixReportIntersection.

Available in all OptiX program types

5.1.2.113 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

5.1.2.114 optixGetRibbonVertexData() [1/2]

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

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

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

Available in AH, CH

5.1.2.115 optixGetRibbonVertexData() [2/2]

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

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

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

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

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

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

Available in all OptiX program types

5.1.2.116 optixGetRibbonVertexDataFromHandle()

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

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

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

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

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

Available in all OptiX program types

5.1.2.117 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

5.1.2.118 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

5.1.2.119 optixGetSphereData() [1/2]

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

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

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

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

Available in AH, CH

5.1.2.120 optixGetSphereData() [2/2]

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

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

optixGetSphereDataFromHandle( ) for random access sphere data.

Returns the object space sphere data, center point and radius, in a Geometry Acceleration Structure (GAS) at a given motion time.

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

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

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

Available in all OptiX program types

5.1.2.121 optixGetSphereDataFromHandle()

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

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

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

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

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

Available in all OptiX program types

5.1.2.122 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

5.1.2.123 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

5.1.2.124 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

5.1.2.125 optixGetTransformListSize()

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

Returns the number of transforms on the current transform list.

Available in IS, AH, CH, EX

5.1.2.126 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

5.1.2.127 optixGetTriangleBarycentrics()

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

Convenience function that returns the first two attributes as floats.

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

Available in AH, CH

5.1.2.128 optixGetTriangleVertexData() [1/2]

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

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

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

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

Available in AH, CH

5.1.2.129 optixGetTriangleVertexData() [2/2]

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

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

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

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

Available in all OptiX program types

5.1.2.130 optixGetTriangleVertexDataFromHandle()

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

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

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

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

Available in all OptiX program types

5.1.2.131 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

5.1.2.132 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

5.1.2.133 optixGetWorldToObjectTransformMatrix() [1/2]

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

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

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

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

Available in IS, AH, CH

5.1.2.134 optixGetWorldToObjectTransformMatrix() [2/2]

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

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

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

Available in IS, AH, CH

5.1.2.135 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

5.1.2.136 optixHitObjectGetAttribute_1()

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

5.1.2.137 optixHitObjectGetAttribute_2()

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

5.1.2.138 optixHitObjectGetAttribute_3()

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

5.1.2.139 optixHitObjectGetAttribute_4()

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

5.1.2.140 optixHitObjectGetAttribute_5()

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

5.1.2.141 optixHitObjectGetAttribute_6()

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

5.1.2.142 optixHitObjectGetAttribute_7()

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

5.1.2.143 optixHitObjectGetCatmullRomRocapsVertexData()

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

5.1.2.144 optixHitObjectGetCatmullRomVertexData()

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

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.145 optixHitObjectGetClusterId()

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

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

Returns OPTIX_CLUSTER_ID_INVALID if a non-Cluster GAS was intersected or if the hit object is a miss.

Available in RG, CH, MS, CC, DC

5.1.2.146 optixHitObjectGetCubicBezierRocapsVertexData()

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

5.1.2.147 optixHitObjectGetCubicBezierVertexData()

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

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.148 optixHitObjectGetCubicBSplineRocapsVertexData()

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

See optixHitObjectGetCubicBSplineVertexData for further documentation.

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

Available in RG, CH, MS, CC, DC

5.1.2.149 optixHitObjectGetCubicBSplineVertexData()

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

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.150 optixHitObjectGetCurveParameter()

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.151 optixHitObjectGetGASTraversableHandle()

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

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

Available in RG, CH, MS, CC, DC

5.1.2.152 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

5.1.2.153 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

5.1.2.154 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

5.1.2.155 optixHitObjectGetLinearCurveVertexData()

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.156 optixHitObjectGetObjectToWorldTransformMatrix()

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.157 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

5.1.2.158 optixHitObjectGetQuadraticBSplineRocapsVertexData()

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

5.1.2.159 optixHitObjectGetQuadraticBSplineVertexData()

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

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

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

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

QUADRATIC_BSPLINE.

Available in RG, CH, MS, CC, DC

5.1.2.160 optixHitObjectGetRayFlags()

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

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

Available in RG, CH, MS, CC, DC

5.1.2.161 optixHitObjectGetRayTime()

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

Returns the rayTime passed into optixTraverse, optixMakeHitObject or optixMakeMissHitObject.

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

Available in RG, CH, MS, CC, DC

5.1.2.162 optixHitObjectGetRayTmax()

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

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

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

Returns 0 for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.163 optixHitObjectGetRayTmin()

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

Returns the tmin passed into optixTraverse, optixMakeHitObject or optixMakeMissHitObject.

Returns 0.0f for nop hit objects.

Available in RG, CH, MS, CC, DC

5.1.2.164 optixHitObjectGetRibbonNormal()

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

Return ribbon normal at intersection reported by optixReportIntersection.

Available in RG, CH, MS, CC, DC

5.1.2.165 optixHitObjectGetRibbonParameters()

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

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

This function is the hit object's equivalent to optixGetRibbonParameters(). It is only valid to call this

function if the return value of optixGetPrimitiveType(optixHitObjectGetHitKind()) equals OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE.

Available in RG, CH, MS, CC, DC

5.1.2.166 optixHitObjectGetRibbonVertexData()

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

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.167 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

5.1.2.168 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

5.1.2.169 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

5.1.2.170 optixHitObjectGetSphereData()

```
static __forceinline__ __device__ void optixHitObjectGetSphereData (
```

```
        float4 data[1] )  [static]
```

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

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

Available in RG, CH, MS, CC, DC

5.1.2.171 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

5.1.2.172 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

5.1.2.173 optixHitObjectGetTraverseData()

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

Serializes the current outgoing hit object which allows to recreate it at a later point using optixMakeHitObject.

Parameters

out	*data*

Available in RG, CH, MS, CC, DC

5.1.2.174 optixHitObjectGetTriangleBarycentrics()

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

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

This function is the hit object's equivalent to optixGetTriangleBarycentrics(). It is only valid to call this function if the return value of optixGetPrimitiveType(optixHitObjectGetHitKind()) equals OPTIX_

PRIMITIVE_TYPE_TRIANGLE.

Available in RG, CH, MS, CC, DC

5.1.2.175 optixHitObjectGetTriangleVertexData()

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.176 optixHitObjectGetWorldRayDirection()

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

Returns the rayDirection passed into optixTraverse, optixMakeHitObject or optixMakeMissHitObject.

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

Available in RG, CH, MS, CC, DC

5.1.2.177 optixHitObjectGetWorldRayOrigin()

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

Returns the rayOrigin passed into optixTraverse, optixMakeHitObject or optixMakeMissHitObject.

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

Available in RG, CH, MS, CC, DC

5.1.2.178 optixHitObjectGetWorldToObjectTransformMatrix()

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

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

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

Available in RG, CH, MS, CC, DC

5.1.2.179 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

5.1.2.180 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

5.1.2.181 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

5.1.2.182 optixHitObjectSetSbtRecordIndex()

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

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

Available in RG, CH, MS, CC

5.1.2.183 optixHitObjectTransformNormalFromObjectToWorldSpace()

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

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

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

Available in IS, AH, CH

5.1.2.184 optixHitObjectTransformNormalFromWorldToObjectSpace()

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

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

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

Available in IS, AH, CH

5.1.2.185 optixHitObjectTransformPointFromObjectToWorldSpace()

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

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

list of the current outgoing hit object.

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

Available in IS, AH, CH

5.1.2.186 optixHitObjectTransformPointFromWorldToObjectSpace()

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

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

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

Available in IS, AH, CH

5.1.2.187 optixHitObjectTransformVectorFromObjectToWorldSpace()

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

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

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

Available in IS, AH, CH

5.1.2.188 optixHitObjectTransformVectorFromWorldToObjectSpace()

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

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

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

Available in IS, AH, CH

5.1.2.189 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

5.1.2.190 optixInvoke() [1/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixInvoke (
          OptixPayloadTypeID type,
```

```
        Payload &... payload ) [static]
```

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

Parameters

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

Available in RG, CH, MS, CC

5.1.2.191 optixInvoke() [2/2]

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

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

Parameters

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

Available in RG, CH, MS, CC

5.1.2.192 optixIsBackFaceHit() [1/2]

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

Function interpreting the hit kind associated with the current optixReportIntersection.

Available in AH, CH

5.1.2.193 optixIsBackFaceHit() [2/2]

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

Function interpreting the result of optixGetHitKind().

Available in all OptiX program types

5.1.2.194 optixIsFrontFaceHit() [1/2]

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

Function interpreting the hit kind associated with the current optixReportIntersection.

Available in AH, CH

5.1.2.195 optixIsFrontFaceHit() [2/2]

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

Function interpreting the result of optixGetHitKind().

Available in all OptiX program types

5.1.2.196 optixIsTriangleBackFaceHit()

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

Convenience function interpreting the result of optixGetHitKind().

Available in AH, CH

5.1.2.197 optixIsTriangleFrontFaceHit()

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

Convenience function interpreting the result of optixGetHitKind().

Available in AH, CH

5.1.2.198 optixIsTriangleHit()

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

Convenience function interpreting the result of optixGetHitKind().

Available in AH, CH

5.1.2.199 optixMakeHitObject()

```
static __forceinline__ __device__ void optixMakeHitObject (
          OptixTraversableHandle handle,
          float3 rayOrigin,
          float3 rayDirection,
          float tmin,
          float rayTime,
          unsigned int rayFlags,
          OptixTraverseData traverseData,
          const OptixTraversableHandle * transforms,
          unsigned int numTransforms ) [static]
```

Constructs an outgoing hit object from the hit object data provided. The traverseData needs to be collected from a previous hit object using optixHitObjectGetTraverseData. This hit object will now become the current outgoing hit object and will overwrite the current outgoing hit object.

Parameters

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

Parameters

in	*rayTime*	
in	*rayFlags*	really only 16 bits, combination of OptixRayFlags
in	*traverseData*	
in	*transforms*	
in	*numTransforms*	

Available in RG, CH, MS, CC

5.1.2.200 optixMakeMissHitObject()

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

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

Parameters

in	*missSBTIndex*	specifies the miss program invoked on a miss
in	*rayOrigin*	
in	*rayDirection*	
in	*tmin*	
in	*tmax*	
in	*rayTime*	
in	*rayFlags*	really only 16 bits, combination of OptixRayFlags

Available in RG, CH, MS, CC

5.1.2.201 optixMakeNopHitObject()

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

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

Available in RG, CH, MS, CC

5.1.2.202 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

5.1.2.203 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

5.1.2.204 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

5.1.2.205 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

5.1.2.206 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

5.1.2.207 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

5.1.2.208 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

5.1.2.209 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

5.1.2.210 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

5.1.2.211 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

5.1.2.212 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

5.1.2.213 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

5.1.2.214 optixSetPayload_1()

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

5.1.2.215 optixSetPayload_10()

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

5.1.2.216 optixSetPayload_11()

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

5.1.2.217 optixSetPayload_12()

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

5.1.2.218 optixSetPayload_13()

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

5.1.2.219 optixSetPayload_14()

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

5.1.2.220 optixSetPayload_15()

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

5.1.2.221 optixSetPayload_16()

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

5.1.2.222 optixSetPayload_17()

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

5.1.2.223 optixSetPayload_18()

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

5.1.2.224 optixSetPayload_19()

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

5.1.2.225 optixSetPayload_2()

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

5.1.2.226 optixSetPayload_20()

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

5.1.2.227 optixSetPayload_21()

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

5.1.2.228 optixSetPayload_22()

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

5.1.2.229 optixSetPayload_23()

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

unsigned int *p*) *[static]*

5.1.2.230 optixSetPayload_24()

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

5.1.2.231 optixSetPayload_25()

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

5.1.2.232 optixSetPayload_26()

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

5.1.2.233 optixSetPayload_27()

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

5.1.2.234 optixSetPayload_28()

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

5.1.2.235 optixSetPayload_29()

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

5.1.2.236 optixSetPayload_3()

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

5.1.2.237 optixSetPayload_30()

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

5.1.2.238 optixSetPayload_31()

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

5.1.2.239 optixSetPayload_4()

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

5.1.2.240 optixSetPayload_5()

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

5.1.2.241 optixSetPayload_6()

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

5.1.2.242 optixSetPayload_7()

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

5.1.2.243 optixSetPayload_8()

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

5.1.2.244 optixSetPayload_9()

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

5.1.2.245 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

5.1.2.246 optixTerminateRay()

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

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

Available in AH

5.1.2.247 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

5.1.2.248 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

5.1.2.249 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

5.1.2.250 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

5.1.2.251 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

5.1.2.252 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

5.1.2.253 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

5.1.2.254 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

5.1.2.255 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

5.1.2.256 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

5.1.2.257 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

5.1.2.258 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

5.1.2.259 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

5.1.2.260 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

5.1.2.261 optixTransformNormalFromObjectToWorldSpace() [1/2]

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

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

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

Available in IS, AH, CH

5.1.2.262 optixTransformNormalFromObjectToWorldSpace() [2/2]

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

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

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

Available in IS, AH, CH

5.1.2.263 optixTransformNormalFromWorldToObjectSpace() [1/2]

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

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

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

Available in IS, AH, CH

5.1.2.264 optixTransformNormalFromWorldToObjectSpace() [2/2]

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

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

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

Available in IS, AH, CH

5.1.2.265 optixTransformPointFromObjectToWorldSpace() [1/2]

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

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

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

Available in IS, AH, CH

5.1.2.266 optixTransformPointFromObjectToWorldSpace() [2/2]

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

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

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

Available in IS, AH, CH

5.1.2.267 optixTransformPointFromWorldToObjectSpace() [1/2]

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

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

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

Available in IS, AH, CH

5.1.2.268 optixTransformPointFromWorldToObjectSpace() [2/2]

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

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

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

Available in IS, AH, CH

5.1.2.269 optixTransformVectorFromObjectToWorldSpace() [1/2]

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

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

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

Available in IS, AH, CH

5.1.2.270 optixTransformVectorFromObjectToWorldSpace() [2/2]

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

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

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

Available in IS, AH, CH

5.1.2.271 optixTransformVectorFromWorldToObjectSpace() [1/2]

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

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

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

Available in IS, AH, CH

5.1.2.272 optixTransformVectorFromWorldToObjectSpace() [2/2]

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

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

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

Available in IS, AH, CH

5.1.2.273 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

Parameters

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

5.1.2.274 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*	
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

5.1.2.275 optixUndefinedValue()

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

Returns an undefined value.

Available anywhere

5.2 Cooperative Vector

Classes

- class OptixCoopVec< T, N >

Functions

- template<typename VecTOut >
  static __forceinline__ __device__ VecTOut optixCoopVecLoad (CUdeviceptr ptr)
- template<typename VecTOut , typename T >
  static __forceinline__ __device__ VecTOut optixCoopVecLoad (T *ptr)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecExp2 (const VecT &vec)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecLog2 (const VecT &vec)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecTanh (const VecT &vec)
- template<typename VecTOut , typename VecTIn >
  static __forceinline__ __device__ VecTOut optixCoopVecCvt (const VecTIn &vec)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecMin (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecMin (const VecT &vecA, typename VecT ::value_type B)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecMax (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecMax (const VecT &vecA, typename VecT ::value_type B)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecMul (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecAdd (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecSub (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecStep (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecFFMA (const VecT &vecA, const VecT &vecB, const VecT &vecC)
- template<typename VecTOut , typename VecTIn , OptixCoopVecElemType inputInterpretation, OptixCoopVecMatrixLayout matrixLayout, bool transpose, unsigned int N, unsigned int K, OptixCoopVecElemType matrixElementType, OptixCoopVecElemType biasElementType>

static __forceinline__ __device__ VecTOut optixCoopVecMatMul (const VecTIn &inputVector, CUdeviceptr matrix, unsigned matrixOffsetInBytes, CUdeviceptr bias, unsigned biasOffsetInBytes, unsigned rowColumnStrideInBytes=0)

- template<typename VecTIn >
  static __forceinline__ __device__ void optixCoopVecReduceSumAccumulate (const VecTIn &inputVector, CUdeviceptr outputVector, unsigned offsetInBytes)
- template<typename VecTA , typename VecTB , OptixCoopVecMatrixLayout matrixLayout = OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL>
  static __forceinline__ __device__ void optixCoopVecOuterProductAccumulate (const VecTA &vecA, const VecTB &vecB, CUdeviceptr outputMatrix, unsigned offsetInBytes, unsigned rowColumnStrideInBytes=0)
- template<unsigned int N, unsigned int K, OptixCoopVecElemType elementType, OptixCoopVecMatrixLayout layout = OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL, unsigned int rowColumnStrideInBytes = 0>
  static __forceinline__ __device__ unsigned int optixCoopVecGetMatrixSize ()

5.2.1 Detailed Description

5.2.2 Function Documentation

5.2.2.1 optixCoopVecAdd()

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

Available anywhere.

5.2.2.2 optixCoopVecCvt()

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

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

Available anywhere

5.2.2.3 optixCoopVecExp2()

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

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

Available anywhere

5.2.2.4 optixCoopVecFFMA()

```
template<typename VecT >
```

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

Available anywhere.

5.2.2.5 optixCoopVecGetMatrixSize()

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

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

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

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

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

Template Parameters

N,K	dimensions of the matrix
elementType	Type of the matrix elements
layout	Layout of the matrix

Available anywhere

5.2.2.6 optixCoopVecLoad() [1/2]

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

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

Available anywhere

5.2.2.7 optixCoopVecLoad() [2/2]

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

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

Available anywhere

5.2.2.8 optixCoopVecLog2()

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

Available anywhere.

5.2.2.9 optixCoopVecMatMul()

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

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

```
         A * B           + C     = D
Does matrix * inputVector + bias  = output
     [NxK]   [Kx1]         [Nx1] = [Nx1]
```

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

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

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

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

When the inputVector's element type does not match the inputInterpretation arithmetically casting is

performed on the input values to match the inputInterpretation.

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

The biasElementType must be specified and compatible even if the pointer supplied is NULL.

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

Template Parameters

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

Parameters

in	*inputVector*	
in	*matrix*	pointer to global memory. Array of NxK elements. 64 byte aligned. Must not be modified during use.
in	*matrixOffsetInBytes*	offset to start of matrix data. Using the same value for matrix with different offsets for all layers yields more effecient execution. 64 byte aligned.
in	*bias*	pointer to global memory. Array of N elements. 16 byte aligned. Must not be modified during use. May be NULL if unused.
in	*biasOffsetInBytes*	offset to start of bias data. Using the same value for bias with different offsets for all layers yields more effecient execution. 16 byte aligned. Ignored if bias is NULL.
in	*rowColumnStrideInBytes*	for row or column major matrix layouts, this identifies the stride between columns or rows.

Available in all OptiX program types

5.2.2.10 optixCoopVecMax() [1/2]

```
template<typename VecT >
static __forceinline__ __device__ VecT optixCoopVecMax (
          const VecT & vecA,
```

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

Available anywhere.

5.2.2.11 optixCoopVecMax() [2/2]

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

Available anywhere.

5.2.2.12 optixCoopVecMin() [1/2]

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

Available anywhere.

5.2.2.13 optixCoopVecMin() [2/2]

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

Available anywhere.

5.2.2.14 optixCoopVecMul()

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

Available anywhere.

5.2.2.15 optixCoopVecOuterProductAccumulate()

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

Produces a matrix outer product of the input vecA and vecB (vecA $*$ transpose(vecB)) and does a component-wise atomic add reduction of the result into global memory starting *offsetInBytes* bytes after

outputMatrix. The dimentions of the matrix are [VecTA::size, VecTB::size]. VecTA::elementType, VecTB ::elementType and the element type of the matrix must be the same, no type conversion is performed. The element type must be OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16.

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

The output matrix will be in matrixLayout layout, though currently only OPTIX_COOP_VEC_ MATRIX_LAYOUT_TRAINING_OPTIMAL layout is supported.

Template Parameters

VecTA	Type of vecA
VecTB	Type of vecB
matrixLayout	Layout of matrix stored in outputMatrix

Parameters

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

Available in all OptiX program types

5.2.2.16 optixCoopVecReduceSumAccumulate()

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

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

VecTIn::elementType must be of type OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16 or OPTIX_COOP_ VEC_ELEM_TYPE_FLOAT32 The memory backed by *outputVector* + *offsetInBytes* must be large enough to accomodate VecTIn::size elements. The type of data in *outputVector* must match VecTIn ::elementType. No type conversion is performed. *outputVector* + *offsetInBytes* must be 4 byte aligned.

Template Parameters

VecTIn	Type of inputVector

Parameters

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

Available in all OptiX program types

5.2.2.17 optixCoopVecStep()

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

Available anywhere.

5.2.2.18 optixCoopVecSub()

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

Available anywhere.

5.2.2.19 optixCoopVecTanh()

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

Available anywhere.

5.3 Function Table

Classes

- struct OptixFunctionTable

Macros

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

Typedefs

- typedef struct OptixFunctionTable OptixFunctionTable

Variables

- OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL

5.3.1 Detailed Description

OptiX Function Table.

5.3.2 Macro Definition Documentation

5.3.2.1 OPTIX_CONCATENATE_ABI_VERSION

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

5.3.2.2 OPTIX_CONCATENATE_ABI_VERSION_IMPL

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

5.3.2.3 OPTIX_FUNCTION_TABLE_SYMBOL

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

5.3.3 Typedef Documentation

5.3.3.1 OptixFunctionTable

```
typedef struct OptixFunctionTable OptixFunctionTable
```

The function table containing all API functions.

See optixInit( ) and optixInitWithHandle( ).

5.3.4 Variable Documentation

5.3.4.1 OPTIX_FUNCTION_TABLE_SYMBOL

```
OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL
```

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

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

5.4 Host API

Modules

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

- Cooperative Vector
- Denoiser

5.4.1 Detailed Description

OptiX Host API.

5.5 Error handling

Functions

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

5.5.1 Detailed Description

5.5.2 Function Documentation

5.5.2.1 optixGetErrorName()

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

Returns a string containing the name of an error code in the enum.

Output is a string representation of the enum. For example "OPTIX_SUCCESS" for OPTIX_SUCCESS and "OPTIX_ERROR_INVALID_VALUE" for OPTIX_ERROR_INVALID_VALUE.

If the error code is not recognized, "Unrecognized OptixResult code" is returned.

Parameters

in	*result*	OptixResult enum to generate string name for

See also optixGetErrorString

5.5.2.2 optixGetErrorString()

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

Returns the description string for an error code.

Output is a string description of the enum. For example "Success" for OPTIX_SUCCESS and "Invalid value" for OPTIX_ERROR_INVALID_VALUE.

If the error code is not recognized, "Unrecognized OptixResult code" is returned.

Parameters

in	*result*	OptixResult enum to generate string description for

See also optixGetErrorName

5.6 Device context

Functions

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

5.6.1 Detailed Description

5.6.2 Function Documentation

5.6.2.1 optixDeviceContextCreate()

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

Create a device context associated with the CUDA context specified with 'fromContext'.

If zero is specified for 'fromContext', OptiX will use the current CUDA context. The CUDA context should be initialized before calling optixDeviceContextCreate.

Parameters

in	*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

5.6.2.2 optixDeviceContextDestroy()

```
OPTIXAPI OptixResult optixDeviceContextDestroy (
          OptixDeviceContext context )
```

Destroys all CPU and GPU state associated with the device.

It will attempt to block on CUDA streams that have launch work outstanding.

Any API objects, such as OptixModule and OptixPipeline, not already destroyed will be destroyed.

Thread safety: A device context must not be destroyed while it is still in use by concurrent API calls in other threads.

5.6.2.3 optixDeviceContextGetCacheDatabaseSizes()

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

Returns the low and high water marks for disk cache garbage collection. If the cache has been disabled by setting the environment variable OPTIX_CACHE_MAXSIZE=0, this function will return 0 for the low and high water marks.

Parameters

in	*context*	the device context
out	*lowWaterMark*	the low water mark
out	*highWaterMark*	the high water mark

5.6.2.4 optixDeviceContextGetCacheEnabled()

```
OPTIXAPI OptixResult optixDeviceContextGetCacheEnabled (
          OptixDeviceContext context,
          int * enabled )
```

Indicates whether the disk cache is enabled or disabled.

Parameters

in	*context*	the device context
out	*enabled*	1 if enabled, 0 if disabled

5.6.2.5 optixDeviceContextGetCacheLocation()

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

Returns the location of the disk cache. If the cache has been disabled by setting the environment variable OPTIX_CACHE_MAXSIZE=0, this function will return an empy string.

Parameters

in	*context*	the device context
out	*location*	directory of disk cache, null terminated if locationSize > 0
in	*locationSize*	locationSize

5.6.2.6 optixDeviceContextGetProperty()

```
OPTIXAPI OptixResult optixDeviceContextGetProperty (
            OptixDeviceContext context,
            OptixDeviceProperty property,
            void * value,
            size_t sizeInBytes )
```

Query properties of a device context.

Parameters

in	*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

5.6.2.7 optixDeviceContextSetCacheDatabaseSizes()

```
OPTIXAPI OptixResult optixDeviceContextSetCacheDatabaseSizes (
            OptixDeviceContext context,
            size_t lowWaterMark,
            size_t highWaterMark )
```

Sets the low and high water marks for disk cache garbage collection.

Garbage collection is triggered when a new entry is written to the cache and the current cache data size plus the size of the cache entry that is about to be inserted exceeds the high water mark. Garbage collection proceeds until the size reaches the low water mark. Garbage collection will always free enough space to insert the new entry without exceeding the low water mark. Setting either limit to zero will disable garbage collection. An error will be returned if both limits are non-zero and the high water mark is smaller than the low water mark.

Note that garbage collection is performed only on writes to the disk cache. No garbage collection is triggered on disk cache initialization or immediately when calling this function, but on subsequent inserting of data into the database.

If the size of a compiled module exceeds the value configured for the high water mark and garbage collection is enabled, the module will not be added to the cache and a warning will be added to the log.

The high water mark can be overridden with the environment variable OPTIX_CACHE_MAXSIZE. The environment variable takes precedence over the function parameters. The low water mark will be set to half the value of OPTIX_CACHE_MAXSIZE. Setting OPTIX_CACHE_MAXSIZE to 0 will disable the disk cache, but will not alter the contents of the cache. Negative and non-integer values will be ignored.

Parameters

in	*context*	the device context
in	*lowWaterMark*	the low water mark
in	*highWaterMark*	the high water mark

5.6.2.8 optixDeviceContextSetCacheEnabled()

```
OPTIXAPI OptixResult optixDeviceContextSetCacheEnabled (
            OptixDeviceContext context,
            int enabled )
```

Enables or disables the disk cache.

If caching was previously disabled, enabling it will attempt to initialize the disk cache database using the currently configured cache location. An error will be returned if initialization fails.

Note that no in-memory cache is used, so no caching behavior will be observed if the disk cache is disabled.

The cache can be disabled by setting the environment variable OPTIX_CACHE_MAXSIZE=0. The environment variable takes precedence over this setting. See optixDeviceContextSetCacheDatabaseSizes for additional information.

Note that the disk cache can be disabled by the environment variable, but it cannot be enabled via the environment if it is disabled via the API.

Parameters

in	*context*	the device context
in	*enabled*	1 to enabled, 0 to disable

5.6.2.9 optixDeviceContextSetCacheLocation()

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

Sets the location of the disk cache.

The location is specified by a directory. This directory should not be used for other purposes and will be created if it does not exist. An error will be returned if 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

5.6.2.10 optixDeviceContextSetLogCallback()

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

Sets the current log callback method.

See OptixLogCallback for more details.

Thread safety: It is guaranteed that the callback itself (callbackFunction and callbackData) are updated atomically. It is not guaranteed that the callback itself (callbackFunction and callbackData) and the callbackLevel are updated atomically. It is unspecified when concurrent API calls using the same context start to make use of the new callback method.

Parameters

in	*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

5.7 Pipelines

Functions

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

5.7.1 Detailed Description

5.7.2 Function Documentation

5.7.2.1 optixPipelineCreate()

```
OPTIXAPI OptixResult optixPipelineCreate (
          OptixDeviceContext context,
```

```
const OptixPipelineCompileOptions * pipelineCompileOptions,
const OptixPipelineLinkOptions * pipelineLinkOptions,
const OptixProgramGroup * programGroups,
unsigned int numProgramGroups,
char * logString,
size_t * logStringSize,
OptixPipeline * pipeline )
```

logString is an optional buffer that contains compiler feedback and errors. This information is also passed to the context logger (if enabled), however it may be difficult to correlate output to the logger to specific API invocations when using multiple threads. The output to logString will only contain feedback for this specific invocation of this API call.

logStringSize as input should be a pointer to the number of bytes backing logString. Upon return it contains the length of the log message (including the null terminator) which may be greater than the input value. In this case, the log message will be truncated to fit into logString.

If logString or logStringSize are NULL, no output is written to logString. If logStringSize points to a value that is zero, no output is written. This does not affect output to the context logger if enabled.

Parameters

in	*context*	
in	*pipelineCompileOptions*	
in	*pipelineLinkOptions*	
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*	

5.7.2.2 optixPipelineDestroy()

```
OPTIXAPI OptixResult optixPipelineDestroy (
          OptixPipeline pipeline )
```

Thread safety: A pipeline must not be destroyed while it is still in use by concurrent API calls in other threads.

5.7.2.3 optixPipelineSetStackSize()

```
OPTIXAPI OptixResult optixPipelineSetStackSize (
          OptixPipeline pipeline,
          unsigned int directCallableStackSizeFromTraversal,
          unsigned int directCallableStackSizeFromState,
          unsigned int continuationStackSize,
          unsigned int maxTraversableGraphDepth )
```

Sets the stack sizes for a pipeline.

Users are encouraged to see the programming guide and the implementations of the helper functions to understand how to construct the stack sizes based on their particular needs.

If this method is not used, an internal default implementation is used. The default implementation is correct (but not necessarily optimal) as long as the maximum depth of call trees of CC programs is at most 2, and no DC programs or motion transforms are used.

The maxTraversableGraphDepth responds to the maximal number of traversables visited when calling trace. Every acceleration structure and motion transform count as one level of traversal. E.g., for a simple IAS (instance acceleration structure) -> GAS (geometry acceleration structure) traversal graph, the maxTraversableGraphDepth is two. For IAS -> MT (motion transform) -> GAS, the maxTraversableGraphDepth is three. Note that it does not matter whether a IAS or GAS has motion or not, it always counts as one. Launching optix with exceptions turned on (see OPTIX_EXCEPTION_FLAG_TRACE_DEPTH) will throw an exception if the specified maxTraversableGraphDepth is too small.

Parameters

in	*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.

5.8 Modules

Functions

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

5.8.1 Detailed Description

5.8.2 Function Documentation

5.8.2.1 optixBuiltinISModuleGet()

```
OPTIXAPI OptixResult optixBuiltinISModuleGet (
```

```
            OptixDeviceContext context,
            const OptixModuleCompileOptions * moduleCompileOptions,
            const OptixPipelineCompileOptions * pipelineCompileOptions,
            const 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.)

5.8.2.2 optixModuleCreate()

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

Compiling programs into a module. These programs can be passed in as either PTX or OptiX-IR.

See the Programming Guide for details, as well as how to generate these encodings from CUDA sources.

logString is an optional buffer that contains compiler feedback and errors. This information is also passed to the context logger (if enabled), however it may be difficult to correlate output to the logger to specific API invocations when using multiple threads. The output to logString will only contain feedback for this specific invocation of this API call.

logStringSize as input should be a pointer to the number of bytes backing logString. Upon return it contains the length of the log message (including the null terminator) which may be greater than the input value. In this case, the log message will be truncated to fit into logString.

If logString or logStringSize are NULL, no output is written to logString. If logStringSize points to a value that is zero, no output is written. This does not affect output to the context logger if enabled.

Parameters

in	*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*	

Parameters

out	*module*	

Returns

OPTIX_ERROR_INVALID_VALUE - context is 0, moduleCompileOptions is 0, pipelineCompileOptions is 0, input is 0, module is 0.

5.8.2.3 optixModuleCreateWithTasks()

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

This function is designed to do just enough work to create the OptixTask return parameter and is expected to be fast enough run without needing parallel execution. A single thread could generate all the OptixTask objects for further processing in a work pool.

Options are similar to optixModuleCreate( ), aside from the return parameter, firstTask.

The memory used to hold the input should be live until all tasks are finished.

It is illegal to call optixModuleDestroy( ) if any OptixTask objects are currently being executed. In that case OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE will be returned.

If an invocation of optixTaskExecute fails, the OptixModule will be marked as OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE if there are outstanding tasks or OPTIX_MODULE_COMPILE_STATE_FAILURE if there are no outstanding tasks. Subsequent calls to optixTaskExecute( ) may execute additional work to collect compilation errors generated from the input. Currently executing tasks will not necessarily be terminated immediately but at the next opportunity.

Logging will continue to be directed to the logger installed with the OptixDeviceContext. If logString is provided to optixModuleCreateWithTasks( ), it will contain all the compiler feedback from all executed tasks. The lifetime of the memory pointed to by logString should extend from calling optixModuleCreateWithTasks( ) to when the compilation state is either OPTIX_MODULE_COMPILE_STATE_FAILURE or OPTIX_MODULE_COMPILE_STATE_COMPLETED. OptiX will not write to the logString outside of execution of optixModuleCreateWithTasks( ) or optixTaskExecute( ). If the compilation state is OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE and no further execution of optixTaskExecute( ) is performed the logString may be reclaimed by the application before calling optixModuleDestroy( ). The contents of logString will contain output from currently completed tasks.

All OptixTask objects associated with a given OptixModule will be cleaned up when optixModuleDestroy( ) is called regardless of whether the compilation was successful or not. If the compilation state is OPTIX_MODULE_COMPILE_STATE_IMPENDIND_FAILURE, any unstarted OptixTask objects do not need to be executed though there is no harm doing so.

See also optixModuleCreate

5.8.2.4 optixModuleDestroy()

```
OPTIXAPI OptixResult optixModuleDestroy (
          OptixModule module )
```

Call for OptixModule objects created with optixModuleCreate and optixModuleDeserialize.

Modules must not be destroyed while they are still used by any program group.

Thread safety: A module must not be destroyed while it is still in use by concurrent API calls in other threads.

5.8.2.5 optixModuleGetCompilationState()

```
OPTIXAPI OptixResult optixModuleGetCompilationState (
          OptixModule module,
          OptixModuleCompileState * state )
```

When creating a module with tasks, the current state of the module can be queried using this function.

Thread safety: Safe to call from any thread until optixModuleDestroy is called.

See also optixModuleCreateWithTasks

5.9 Tasks

Functions

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

5.9.1 Detailed Description

5.9.2 Function Documentation

5.9.2.1 optixTaskExecute()

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

Each OptixTask should be executed with optixTaskExecute( ). If additional parallel work is found, new OptixTask objects will be returned in additionalTasks along with the number of additional tasks in numAdditionalTasksCreated. The parameter additionalTasks should point to a user allocated array of minimum size maxNumAdditionalTasks. OptiX can generate upto maxNumAdditionalTasks additional tasks.

Each task can be executed in parallel and in any order.

Thread safety: Safe to call from any thread until optixModuleDestroy( ) is called for any associated task.

See also optixModuleCreateWithTasks

Parameters

in	*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

5.10 Program groups

Functions

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

5.10.1 Detailed Description

5.10.2 Function Documentation

5.10.2.1 optixProgramGroupCreate()

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

logString is an optional buffer that contains compiler feedback and errors. This information is also passed to the context logger (if enabled), however it may be difficult to correlate output to the logger to specific API invocations when using multiple threads. The output to logString will only contain feedback for this specific invocation of this API call.

logStringSize as input should be a pointer to the number of bytes backing logString. Upon return it contains the length of the log message (including the null terminator) which may be greater than the input value. In this case, the log message will be truncated to fit into logString.

If logString or logStringSize are NULL, no output is written to logString. If logStringSize points to a value that is zero, no output is written. This does not affect output to the context logger if enabled.

Creates numProgramGroups OptiXProgramGroup objects from the specified OptixProgramGroupDesc array. The size of the arrays must match.

Parameters

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

5.10.2.2 optixProgramGroupDestroy()

```
OPTIXAPI OptixResult optixProgramGroupDestroy (
          OptixProgramGroup programGroup )
```

Thread safety: A program group must not be destroyed while it is still in use by concurrent API calls in other threads.

5.10.2.3 optixProgramGroupGetStackSize()

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

Returns the stack sizes for the given program group. When programs in this `programGroup` are relying on external functions, the corresponding stack sizes can only be correctly retrieved when all functions are known after linking, i.e. when a pipeline has been created. When `pipeline` is set to NULL, the stack size will be calculated excluding external functions. In this case a warning will be issued if external functions are referenced by the OptixModule.

Parameters

in	*programGroup*	the program group
out	*stackSizes*	the corresponding stack sizes
in	*pipeline*	considering the program group within the given pipeline, can be NULL

5.10.2.4 optixSbtRecordPackHeader()

```
OPTIXAPI OptixResult optixSbtRecordPackHeader (
          OptixProgramGroup programGroup,
          void * sbtRecordHeaderHostPointer )
```

Parameters

in	*programGroup*	the program group containing the program(s)
out	*sbtRecordHeaderHostPointer*	the result sbt record header

5.11 Launches

Functions

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

5.11.1 Detailed Description

5.11.2 Function Documentation

5.11.2.1 optixLaunch()

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

Where the magic happens.

The stream and pipeline must belong to the same device context. Multiple launches may be issues in parallel from multiple threads to different streams.

pipelineParamsSize number of bytes are copied from the device memory pointed to by pipelineParams before launch. It is an error if pipelineParamsSize is greater than the size of the variable declared in modules and identified by OptixPipelineCompileOptions::pipelineLaunchParamsVariableName. If the launch params variable was optimized out or not found in the modules linked to the pipeline then the pipelineParams and pipelineParamsSize parameters are ignored.

sbt points to the shader binding table, which defines shader groupings and their resources. See the SBT spec.

Parameters

in	*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.

5.12 Acceleration structures

Functions

- OPTIXAPI OptixResult optixAccelComputeMemoryUsage (OptixDeviceContext context, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, OptixAccelBufferSizes *bufferSizes)
- OPTIXAPI OptixResult optixAccelBuild (OptixDeviceContext context, CUstream stream, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, CUdeviceptr tempBuffer, size_t tempBufferSizeInBytes, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle, const OptixAccelEmitDesc *emittedProperties, unsigned int numEmittedProperties)
- OPTIXAPI OptixResult optixAccelGetRelocationInfo (OptixDeviceContext context, OptixTraversableHandle handle, OptixRelocationInfo *info)
- OPTIXAPI OptixResult optixCheckRelocationCompatibility (OptixDeviceContext context, const OptixRelocationInfo *info, int *compatible)
- OPTIXAPI OptixResult optixAccelRelocate (OptixDeviceContext context, CUstream stream, const OptixRelocationInfo *info, const OptixRelocateInput *relocateInputs, size_t numRelocateInputs, CUdeviceptr targetAccel, size_t targetAccelSizeInBytes, OptixTraversableHandle *targetHandle)
- OPTIXAPI OptixResult optixAccelCompact (OptixDeviceContext context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle)
- OPTIXAPI OptixResult optixAccelEmitProperty (OptixDeviceContext context, CUstream stream, OptixTraversableHandle handle, const OptixAccelEmitDesc *emittedProperty)
- OPTIXAPI OptixResult optixConvertPointerToTraversableHandle (OptixDeviceContext onDevice, CUdeviceptr pointer, OptixTraversableType traversableType, OptixTraversableHandle *traversableHandle)
- OPTIXAPI OptixResult optixOpacityMicromapArrayComputeMemoryUsage (OptixDeviceContext context, const OptixOpacityMicromapArrayBuildInput *buildInput, OptixMicromapBufferSizes *bufferSizes)
- OPTIXAPI OptixResult optixOpacityMicromapArrayBuild (OptixDeviceContext context, CUstream stream, const OptixOpacityMicromapArrayBuildInput *buildInput, const OptixMicromapBuffers *buffers)
- OPTIXAPI OptixResult optixOpacityMicromapArrayGetRelocationInfo (OptixDeviceContext context, CUdeviceptr opacityMicromapArray, OptixRelocationInfo *info)
- OPTIXAPI OptixResult optixOpacityMicromapArrayRelocate (OptixDeviceContext context, CUstream stream, const OptixRelocationInfo *info, CUdeviceptr targetOpacityMicromapArray, size_t targetOpacityMicromapArraySizeInBytes)
- OPTIXAPI OptixResult optixClusterAccelComputeMemoryUsage (OptixDeviceContext context, OptixClusterAccelBuildMode buildMode, const OptixClusterAccelBuildInput *buildInput, OptixAccelBufferSizes *bufferSizes)
- OPTIXAPI OptixResult optixClusterAccelBuild (OptixDeviceContext context, CUstream stream, const OptixClusterAccelBuildModeDesc *buildModeDesc, const OptixClusterAccelBuildInput *buildInput, CUdeviceptr argsArray, CUdeviceptr argsCount, unsigned int argsStrideInBytes)

5.12.1 Detailed Description

5.12.2 Function Documentation

5.12.2.1 optixAccelBuild()

```
OPTIXAPI OptixResult optixAccelBuild (
          OptixDeviceContext context,
```

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

Parameters

in	*context*	
in	*stream*	
in	*accelOptions*	accel options
in	*buildInputs*	an array of OptixBuildInput objects
in	*numBuildInputs*	must be >= 1 for GAS, and == 1 for IAS
in	*tempBuffer*	must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT
in	*tempBufferSizeInBytes*	
in	*outputBuffer*	must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT
in	*outputBufferSizeInBytes*	
out	*outputHandle*	
in	*emittedProperties*	types of requested properties and output buffers
in	*numEmittedProperties*	number of post-build properties to populate (may be zero)

5.12.2.2 optixAccelCompact()

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

After building an acceleration structure, it can be copied in a compacted form to reduce memory. In order to be compacted, OPTIX_BUILD_FLAG_ALLOW_COMPACTION must be supplied in OptixAccelBuildOptions::buildFlags passed to optixAccelBuild.

'outputBuffer' is the pointer to where the compacted acceleration structure will be written. This pointer must be a multiple of OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT.

The size of the memory specified in 'outputBufferSizeInBytes' should be at least the value computed using the OPTIX_PROPERTY_TYPE_COMPACTED_SIZE that was reported during optixAccelBuild.

Parameters

in	*context*
in	*stream*
in	*inputHandle*
in	*outputBuffer*
in	*outputBufferSizeInBytes*
out	*outputHandle*

5.12.2.3 optixAccelComputeMemoryUsage()

```
OPTIXAPI 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

5.12.2.4 optixAccelEmitProperty()

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

Emit a single property after an acceleration structure was built. The result buffer of the ' emittedProperty' needs to be large enough to hold the requested property (.

See also OptixAccelPropertyType).

Parameters

in	*context*	
in	*stream*	
in	*handle*	
in	*emittedProperty*	type of requested property and output buffer

5.12.2.5 optixAccelGetRelocationInfo()

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

Obtain relocation information, stored in OptixRelocationInfo, for a given context and acceleration structure's traversable handle.

The relocation information can be passed to optixCheckRelocationCompatibility to determine if an acceleration structure, referenced by 'handle', can be relocated to a different device's memory space (see optixCheckRelocationCompatibility).

When used with optixAccelRelocate, it provides data necessary for doing the relocation.

If the acceleration structure data associated with 'handle' is copied multiple times, the same OptixRelocationInfo can also be used on all copies.

Parameters

in	*context*
in	*handle*
out	*info*

Returns

OPTIX_ERROR_INVALID_VALUE will be returned for traversable handles that are not from acceleration structure builds.

5.12.2.6 optixAccelRelocate()

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

optixAccelRelocate is called to update the acceleration structure after it has been relocated. Relocation is necessary when the acceleration structure's location in device memory has changed. optixAccelRelocate does not copy the memory. This function only operates on the relocated memory whose new location is specified by 'targetAccel'. optixAccelRelocate also returns the new OptixTraversableHandle associated with 'targetAccel'. The original memory (source) is not required to be valid, only the OptixRelocationInfo.

Before calling optixAccelRelocate, optixCheckRelocationCompatibility should be called to ensure the copy will be compatible with the destination device context.

The memory pointed to by 'targetAccel' should be allocated with the same size as the source acceleration. Similar to the 'outputBuffer' used in optixAccelBuild, this pointer must be a multiple of

OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT.

The memory in 'targetAccel' must be allocated as long as the accel is in use.

The instance traversables referenced by an IAS and the micromaps referenced by a triangle GAS may themselves require relocation. 'relocateInputs' and 'numRelocateInputs' should be used to specify the relocated traversables and micromaps. After relocation, the relocated accel will reference these relocated traversables and micromaps instead of their sources. The number of relocate inputs 'numRelocateInputs' must match the number of build inputs 'numBuildInputs' used to build the source accel. Relocation inputs correspond with build inputs used to build the source accel and should appear in the same order (see optixAccelBuild). 'relocateInputs' and 'numRelocateInputs' may be zero, preserving any references to traversables and micromaps from the source accel.

Parameters

in	*context*
in	*stream*
in	*info*
in	*relocateInputs*
in	*numRelocateInputs*
in	*targetAccel*
in	*targetAccelSizeInBytes*
out	*targetHandle*

5.12.2.7 optixCheckRelocationCompatibility()

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

Checks if an optix data structure built using another OptixDeviceContext (that was used to fill in 'info') is compatible with the OptixDeviceContext specified in the 'context' parameter.

Any device is always compatible with itself.

Parameters

in	*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.

5.12.2.8 optixClusterAccelBuild()

```
OPTIXAPI OptixResult optixClusterAccelBuild (
          OptixDeviceContext context,
```

```
CUstream stream,
const OptixClusterAccelBuildModeDesc * buildModeDesc,
const OptixClusterAccelBuildInput * buildInput,
CUdeviceptr argsArray,
CUdeviceptr argsCount,
unsigned int argsStrideInBytes )
```

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

Parameters

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

5.12.2.9 optixClusterAccelComputeMemoryUsage()

```
OPTIXAPI OptixResult optixClusterAccelComputeMemoryUsage (
        OptixDeviceContext context,
        OptixClusterAccelBuildMode buildMode,
```

```
        const OptixClusterAccelBuildInput * buildInput,
        OptixAccelBufferSizes * bufferSizes )
```

Host side conservative memory computation for a subsequent optixClusterAccelBuild call with the same build mode and input. For implicit builds, the output buffer size contains the required size for holding all build outputs as specified in buildInput->maxArgsCount. For explicit builds, the output buffer size contains the required size for holding a single build output. The temp buffer of any optixClusterAccelBuild call must be at least as big as reported by optixClusterAccelComputeMemoryUsage. optixClusterAccelComputeMemoryUsage always returns 0 for OptixAccelBufferSizes::tempUpdateSizeInBytes.

Parameters

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

5.12.2.10 optixConvertPointerToTraversableHandle()

```
OPTIXAPI 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

5.12.2.11 optixOpacityMicromapArrayBuild()

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

Construct an array of Opacity Micromaps.

Each triangle within an instance/GAS may reference one opacity micromap to give finer control over alpha behavior. A opacity micromap consists of a set of 4^N micro-triangles in a triangular uniform barycentric grid. Multiple opacity micromaps are collected (built) into a opacity micromap array with this function. Each geometry in a GAS may bind a single opacity micromap array and can use opacity

micromaps from that array only.

Each micro-triangle within a opacity micromap can be in one of four states: Transparent, Opaque, Unknown-Transparent or Unknown-Opaque. During traversal, if a triangle with a opacity micromap attached is intersected, the opacity micromap is queried to categorize the hit as either opaque, unknown (alpha) or a miss. Geometry, ray or instance flags that modify the alpha/opaque behavior are applied *after* this opacity micromap query.

The opacity micromap query may operate in 2-state mode (alpha testing) or 4-state mode (AHS culling), depending on the opacity micromap type and ray/instance flags. When operating in 2-state mode, alpha hits will not be reported, and transparent and opaque hits must be accurate.

Parameters

in	*context*	
in	*stream*	
in	*buildInput*	a single build input object referencing many opacity micromaps
in	*buffers*	the buffers used for build

5.12.2.12 optixOpacityMicromapArrayComputeMemoryUsage()

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

Determine the amount of memory necessary for a Opacity Micromap Array build.

Parameters

in	*context*
in	*buildInput*
out	*bufferSizes*

5.12.2.13 optixOpacityMicromapArrayGetRelocationInfo()

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

Obtain relocation information, stored in OptixRelocationInfo, for a given context and opacity micromap array.

The relocation information can be passed to optixCheckRelocationCompatibility to determine if a opacity micromap array, referenced by buffers, can be relocated to a different device's memory space (see optixCheckRelocationCompatibility).

When used with optixOpacityMicromapArrayRelocate, it provides data necessary for doing the relocation.

If the opacity micromap array data associated with 'opacityMicromapArray' is copied multiple times, the same OptixRelocationInfo can also be used on all copies.

Parameters

in	*context*
in	*opacityMicromapArray*
out	*info*

5.12.2.14 optixOpacityMicromapArrayRelocate()

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

optixOpacityMicromapArrayRelocate is called to update the opacity micromap array after it has been relocated. Relocation is necessary when the opacity micromap array's location in device memory has changed. optixOpacityMicromapArrayRelocate does not copy the memory. This function only operates on the relocated memory whose new location is specified by 'targetOpacityMicromapArray'. The original memory (source) is not required to be valid, only the OptixRelocationInfo.

Before calling optixOpacityMicromapArrayRelocate, optixCheckRelocationCompatibility should be called to ensure the copy will be compatible with the destination device context.

The memory pointed to by 'targetOpacityMicromapArray' should be allocated with the same size as the source opacity micromap array. Similar to the 'OptixMicromapBuffers::output' used in optixOpacityMicromapArrayBuild, this pointer must be a multiple of OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT.

The memory in 'targetOpacityMicromapArray' must be allocated as long as the opacity micromap array is in use.

Note that any Acceleration Structures build using the original memory (source) as input will still be associated with this original memory. To associate an existing (possibly relocated) Acceleration Structures with the relocated opacity micromap array, use optixAccelBuild to update the existing Acceleration Structures (See OPTIX_BUILD_OPERATION_UPDATE)

Parameters

in	*context*
in	*stream*
in	*info*
in	*targetOpacityMicromapArray*
in	*targetOpacityMicromapArraySizeInBytes*

5.13 Cooperative Vector

Functions

- OPTIXAPI OptixResult optixCoopVecMatrixConvert (OptixDeviceContext context, CUstream stream, unsigned int numNetworks, const OptixNetworkDescription *inputNetworkDescription, CUdeviceptr inputNetworks, size_t inputNetworkStrideInBytes, const OptixNetworkDescription

∗outputNetworkDescription, CUdeviceptr outputNetworks, size_t outputNetworkStrideInBytes)

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

Variables

- int pointerType

5.13.1 Detailed Description

5.13.2 Function Documentation

5.13.2.1 optixCoopVecMatrixComputeSize()

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

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

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

Parameters

in	*context*	
in	*elementType*	
in	*N*	
in	*K*	
in	*layout*	
in	*rowColumnStrideInBytes*	Ignored for optimal layouts
out	*sizeInBytes*	Output size of the matrix in bytes

5.13.2.2 optixCoopVecMatrixConvert()

```
OPTIXAPI OptixResult optixCoopVecMatrixConvert (
            OptixDeviceContext context,
            CUstream stream,
            unsigned int numNetworks,
            const OptixNetworkDescription * inputNetworkDescription,
            CUdeviceptr inputNetworks,
```

```
    size_t inputNetworkStrideInBytes,
    const OptixNetworkDescription * outputNetworkDescription,
    CUdeviceptr outputNetworks,
    size_t outputNetworkStrideInBytes )
```

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

One use case is to convert a matrix in OPTIX_COOP_VEC_MATRIX_LAYOUT_ROW_MAJOR or OPTIX_COOP_VEC_MATRIX_LAYOUT_COLUMN_MAJOR into OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL.

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

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

Parameters

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

5.13.3 Variable Documentation

5.13.3.1 pointerType

```
int pointerType
```

5.14 Denoiser

Functions

- OPTIXAPI OptixResult optixDenoiserCreate (OptixDeviceContext context, OptixDenoiserModelKind modelKind, const OptixDenoiserOptions *options, OptixDenoiser *denoiser)
- OPTIXAPI OptixResult optixDenoiserCreateWithUserModel (OptixDeviceContext context, const

void *userData, size_t userDataSizeInBytes, OptixDenoiser *denoiser)
- OPTIXAPI OptixResult optixDenoiserDestroy (OptixDenoiser denoiser)
- OPTIXAPI OptixResult optixDenoiserComputeMemoryResources (const OptixDenoiser denoiser, unsigned int outputWidth, unsigned int outputHeight, OptixDenoiserSizes *returnSizes)
- OPTIXAPI OptixResult optixDenoiserSetup (OptixDenoiser denoiser, CUstream stream, unsigned int inputWidth, unsigned int inputHeight, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OPTIXAPI OptixResult optixDenoiserInvoke (OptixDenoiser denoiser, CUstream stream, const OptixDenoiserParams *params, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, const OptixDenoiserGuideLayer *guideLayer, const OptixDenoiserLayer *layers, unsigned int numLayers, unsigned int inputOffsetX, unsigned int inputOffsetY, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OPTIXAPI OptixResult optixDenoiserComputeIntensity (OptixDenoiser denoiser, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputIntensity, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OPTIXAPI OptixResult optixDenoiserComputeAverageColor (OptixDenoiser denoiser, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputAverageColor, CUdeviceptr scratch, size_t scratchSizeInBytes)

5.14.1 Detailed Description

5.14.2 Function Documentation

5.14.2.1 optixDenoiserComputeAverageColor()

```
OPTIXAPI OptixResult optixDenoiserComputeAverageColor (
            OptixDenoiser denoiser,
            CUstream stream,
            const OptixImage2D * inputImage,
            CUdeviceptr outputAverageColor,
            CUdeviceptr scratch,
            size_t scratchSizeInBytes )
```

Compute average logarithmic for each of the first three channels for the given image. When denoising tiles the intensity of the entire image should be computed, i.e. not per tile to get consistent results.

The size of scratch memory required can be queried with optixDenoiserComputeMemoryResources.

data type unsigned char is not supported for 'inputImage', it must be 3 or 4 component half/float.

Parameters

in	*denoiser*	
in	*stream*	
in	*inputImage*	
out	*outputAverageColor*	three floats
in	*scratch*	
in	*scratchSizeInBytes*	

5.14.2.2 optixDenoiserComputeIntensity()

```
OPTIXAPI OptixResult optixDenoiserComputeIntensity (
            OptixDenoiser denoiser,
            CUstream stream,
            const OptixImage2D * inputImage,
            CUdeviceptr outputIntensity,
            CUdeviceptr scratch,
            size_t scratchSizeInBytes )
```

Computes the logarithmic average intensity of the given image. The returned value 'outputIntensity' is multiplied with the RGB values of the input image/tile in optixDenoiserInvoke if given in the parameter OptixDenoiserParams::hdrIntensity (otherwise 'hdrIntensity' must be a null pointer). This is useful for denoising HDR images which are very dark or bright. When denoising tiles the intensity of the entire image should be computed, i.e. not per tile to get consistent results.

For each RGB pixel in the inputImage the intensity is calculated and summed if it is greater than 1e-8f: 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*	

5.14.2.3 optixDenoiserComputeMemoryResources()

```
OPTIXAPI OptixResult optixDenoiserComputeMemoryResources (
            const OptixDenoiser denoiser,
            unsigned int outputWidth,
            unsigned int outputHeight,
            OptixDenoiserSizes * returnSizes )
```

Computes the GPU memory resources required to execute the denoiser.

Memory for state and scratch buffers must be allocated with the sizes in 'returnSizes' and scratch memory passed to optixDenoiserSetup, optixDenoiserInvoke, optixDenoiserComputeIntensity and optixDenoiserComputeAverageColor. For tiled denoising an overlap area ('overlapWindowSizeInPixels') must be added to each tile on all sides which increases the amount of memory needed to denoise a tile. In case of tiling use withOverlapScratchSizeInBytes for scratch memory size. If only full resolution images are denoised, withoutOverlapScratchSizeInBytes can be used which is always smaller than withOverlapScratchSizeInBytes.

'outputWidth' and 'outputHeight' is the dimension of the image to be denoised (without overlap in case tiling is being used). 'outputWidth' and 'outputHeight' must be greater than or equal to the dimensions passed to optixDenoiserSetup.

Parameters

in	*denoiser*
in	*outputWidth*
in	*outputHeight*
out	*returnSizes*

5.14.2.4 optixDenoiserCreate()

```
OPTIXAPI OptixResult optixDenoiserCreate (
            OptixDeviceContext context,
            OptixDenoiserModelKind modelKind,
            const OptixDenoiserOptions * options,
            OptixDenoiser * denoiser )
```

Creates a denoiser object with the given options, using built-in inference models.

'modelKind' selects the model used for inference. Inference for the built-in models can be guided (giving hints to improve image quality) with albedo and normal vector images in the guide layer (see 'optixDenoiserInvoke'). Use of these images must be enabled in 'OptixDenoiserOptions'.

Parameters

in	*context*
in	*modelKind*
in	*options*
out	*denoiser*

5.14.2.5 optixDenoiserCreateWithUserModel()

```
OPTIXAPI OptixResult optixDenoiserCreateWithUserModel (
            OptixDeviceContext context,
            const void * userData,
            size_t userDataSizeInBytes,
            OptixDenoiser * denoiser )
```

Creates a denoiser object with the given options, using a provided inference model.

'userData' and 'userDataSizeInBytes' provide a user model for inference. The memory passed in userData will be accessed only during the invocation of this function and can be freed after it returns. The user model must export only one weight set which determines both the model kind and the required set of guide images.

Parameters

in	*context*

Parameters

in	*userData*
in	*userDataSizeInBytes*
out	*denoiser*

5.14.2.6 optixDenoiserDestroy()

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

Destroys the denoiser object and any associated host resources.

5.14.2.7 optixDenoiserInvoke()

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

Invokes denoiser on a set of input data and produces at least one output image. State memory must be available during the execution of the denoiser (or until optixDenoiserSetup is called with a new state memory pointer). Scratch memory passed is used only for the duration of this function. Scratch and state memory sizes must have a size greater than or equal to the sizes as returned by optixDenoiserComputeMemoryResources.

'inputOffsetX' and 'inputOffsetY' are pixel offsets in the 'inputLayers' image specifying the beginning of the image without overlap. When denoising an entire image without tiling there is no overlap and 'inputOffsetX' and 'inputOffsetY' must be zero. When denoising a tile which is adjacent to one of the four sides of the entire image the corresponding offsets must also be zero since there is no overlap at the side adjacent to the image border.

'guideLayer' provides additional information to the denoiser. When providing albedo and normal vector guide images, the corresponding fields in the 'OptixDenoiserOptions' must be enabled, see optixDenoiserCreate. 'guideLayer' must not be null. If a guide image in 'OptixDenoiserOptions' is not enabled, the corresponding image in 'OptixDenoiserGuideLayer' is ignored.

If OPTIX_DENOISER_MODEL_KIND_TEMPORAL or OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV is selected, a 2d flow image must be given in 'OptixDenoiserGuideLayer'. It describes for each pixel the flow from the previous to the current frame (a 2d vector in pixel space). The denoised beauty/AOV of the previous frame must be given in 'previousOutput'. If this image is not available in the first frame of a sequence, the noisy beauty/AOV from the first frame and zero flow vectors could be given as a substitute. For non-temporal model kinds the flow image in

'OptixDenoiserGuideLayer' is ignored. 'previousOutput' and 'output' may refer to the same buffer if tiling is not used, i.e. 'previousOutput' is first read by this function and later overwritten with the denoised result. 'output' can be passed as 'previousOutput' to the next frame. In other model kinds (not temporal) 'previousOutput' is ignored.

The beauty layer must be given as the first entry in 'layers'. In AOV type model kinds (OPTIX_DENOISER_MODEL_KIND_AOV or in user defined models implementing kernel-prediction) additional layers for the AOV images can be given. In each layer the noisy input image is given in 'input', the denoised output is written into the 'output' image. input and output images may refer to the same buffer, with the restriction that the pixel formats must be identical for input and output when the blend mode is selected (see OptixDenoiserParams).

If OPTIX_DENOISER_MODEL_KIND_TEMPORAL or OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV is selected, the denoised image from the previous frame must be given in 'previousOutput' in the layer. 'previousOutput' and 'output' may refer to the same buffer if tiling is not used, i.e. 'previousOutput' is first read by this function and later overwritten with the denoised result. 'output' can be passed as 'previousOutput' to the next frame. In addition, 'previousOutputInternalGuideLayer' and 'outputInternalGuideLayer' must both be allocated regardless of tiling mode. The pixel format must be OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER and the dimension must be identical to 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*

5.14.2.8 optixDenoiserSetup()

```
OPTIXAPI OptixResult optixDenoiserSetup (
          OptixDenoiser denoiser,
          CUstream stream,
          unsigned int inputWidth,
          unsigned int inputHeight,
```

```
CUdeviceptr denoiserState,
size_t denoiserStateSizeInBytes,
CUdeviceptr scratch,
size_t scratchSizeInBytes )
```

Initializes the state required by the denoiser.

'inputWidth' and 'inputHeight' must include overlap on both sides of the image if tiling is being used. The overlap is returned by optixDenoiserComputeMemoryResources. For subsequent calls to optixDenoiserInvoke 'inputWidth' and 'inputHeight' are the maximum dimensions of the input layers. Dimensions of the input layers passed to optixDenoiserInvoke may be different in each invocation however they always must be smaller than 'inputWidth' and 'inputHeight' passed to optixDenoiserSetup.

Parameters

in	*denoiser*
in	*stream*
in	*inputWidth*
in	*inputHeight*
in	*denoiserState*
in	*denoiserStateSizeInBytes*
in	*scratch*
in	*scratchSizeInBytes*

5.15 Utilities

Classes

- struct OptixUtilDenoiserImageTile

Macros

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

Functions

- OPTIX_MICROMAP_INLINE_FUNC float optix_impl::__uint_as_float (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::extractEvenBits (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::prefixEor (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC void optix_impl::index2dbary (unsigned int index, unsigned int &u, unsigned int &v, unsigned int &w)
- OPTIX_MICROMAP_INLINE_FUNC void optix_impl::micro2bary (unsigned int index, unsigned int subdivisionLevel, float2 &bary0, float2 &bary1, float2 &bary2)
- OPTIX_MICROMAP_INLINE_FUNC float2 optix_impl::base2micro (const float2 &baseBarycentrics, const float2 microVertexBaseBarycentrics[3])
- OptixResult optixUtilGetPixelStride (const OptixImage2D &image, unsigned int &pixelStrideInBytes)
- OptixResult optixUtilDenoiserSplitImage (const OptixImage2D &input, const OptixImage2D &output, unsigned int overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight, std::vector< OptixUtilDenoiserImageTile > &tiles)

- OptixResult optixUtilDenoiserInvokeTiled (OptixDenoiser denoiser, CUstream stream, const OptixDenoiserParams *params, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, const OptixDenoiserGuideLayer *guideLayer, const OptixDenoiserLayer *layers, unsigned int numLayers, CUdeviceptr scratch, size_t scratchSizeInBytes, unsigned int overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight)
- OptixResult optixUtilAccumulateStackSizes (OptixProgramGroup programGroup, OptixStackSizes *stackSizes, OptixPipeline pipeline)
- OptixResult optixUtilComputeStackSizes (const OptixStackSizes *stackSizes, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepth, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesDCSplit (const OptixStackSizes *stackSizes, unsigned int dssDCFromTraversal, unsigned int dssDCFromState, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepthFromTraversal, unsigned int maxDCDepthFromState, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesCssCCTree (const OptixStackSizes *stackSizes, unsigned int cssCCTree, unsigned int maxTraceDepth, unsigned int maxDCDepth, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesSimplePathTracer (OptixProgramGroup programGroupRG, OptixProgramGroup programGroupMS1, const OptixProgramGroup *programGroupCH1, unsigned int programGroupCH1Count, OptixProgramGroup programGroupMS2, const OptixProgramGroup *programGroupCH2, unsigned int programGroupCH2Count, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize, OptixPipeline pipeline)
- OPTIXAPI OptixResult optixInitWithHandle (void **handlePtr)
- OPTIXAPI OptixResult optixInit (void)
- OPTIXAPI OptixResult optixUninitWithHandle (void *handle)

5.15.1 Detailed Description

OptiX Utilities.

5.15.2 Macro Definition Documentation

5.15.2.1 OPTIX_MICROMAP_FLOAT2_SUB

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

5.15.2.2 OPTIX_MICROMAP_INLINE_FUNC

```
#define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline
```

5.15.3 Function Documentation

5.15.3.1 __uint_as_float()

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

5.15.3.2 base2micro()

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

5.15.3.3 extractEvenBits()

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

5.15.3.4 index2dbary()

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

5.15.3.5 micro2bary()

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

5.15.3.6 optixInit()

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

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

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

5.15.3.7 optixInitWithHandle()

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

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

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

See also optixUninitWithHandle

5.15.3.8 optixUninitWithHandle()

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

Unloads the OptiX library and zeros the function table used by the stubs below. Takes the handle

returned by optixInitWithHandle. All OptixDeviceContext objects must be destroyed before calling this function, or the behavior is undefined.

See also optixInitWithHandle

5.15.3.9 optixUtilAccumulateStackSizes()

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

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

5.15.3.10 optixUtilComputeStackSizes()

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

Computes the stack size values needed to configure a pipeline.

See the programming guide for an explanation of the formula.

Parameters

in	*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.

5.15.3.11 optixUtilComputeStackSizesCssCCTree()

```
OptixResult optixUtilComputeStackSizesCssCCTree (
```

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

Computes the stack size values needed to configure a pipeline.

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

See programming guide for an explanation of the formula.

Parameters

in	*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.

5.15.3.12 optixUtilComputeStackSizesDCSplit( )

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

Computes the stack size values needed to configure a pipeline.

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

See programming guide for an explanation of the formula.

Parameters

in	*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.

5.15.3.13 optixUtilComputeStackSizesSimplePathTracer( )

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

Computes the stack size values needed to configure a pipeline.

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

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

See programming guide for an explanation of the formula.

If the programs rely on external functions, passing the current pipeline will consider these as well.

Otherwise, a null pointer can be passed instead. When external functions are present, a warning will be issued for these cases.

5.15.3.14 optixUtilDenoiserInvokeTiled()

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

Run denoiser on input layers see optixDenoiserInvoke additional parameters:

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

Parameters

in	*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*

5.15.3.15 optixUtilDenoiserSplitImage()

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

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

Parameters

in	*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

5.15.3.16 optixUtilGetPixelStride()

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

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

Parameters

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

5.15.3.17 prefixEor()

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

5.16 Types

Classes

- struct OptixDeviceContextOptions
- struct OptixOpacityMicromapUsageCount
- struct OptixBuildInputOpacityMicromap
- struct OptixRelocateInputOpacityMicromap
- struct OptixBuildInputTriangleArray
- struct OptixRelocateInputTriangleArray

- struct OptixBuildInputCurveArray
- struct OptixBuildInputSphereArray
- struct OptixAabb
- struct OptixBuildInputCustomPrimitiveArray
- struct OptixBuildInputInstanceArray
- struct OptixRelocateInputInstanceArray
- struct OptixBuildInput
- struct OptixRelocateInput
- struct OptixInstance
- struct OptixOpacityMicromapDesc
- struct OptixOpacityMicromapHistogramEntry
- struct OptixOpacityMicromapArrayBuildInput
- struct OptixMicromapBufferSizes
- struct OptixMicromapBuffers
- struct OptixMotionOptions
- struct OptixAccelBuildOptions
- struct OptixAccelBufferSizes
- struct OptixAccelEmitDesc
- struct OptixRelocationInfo
- struct OptixStaticTransform
- struct OptixMatrixMotionTransform
- struct OptixSRTData
- struct OptixSRTMotionTransform
- struct OptixClusterAccelBuildModeDescImplicitDest
- struct OptixClusterAccelBuildModeDescExplicitDest
- struct OptixClusterAccelBuildModeDescGetSize
- struct OptixClusterAccelBuildInputTriangles
- struct OptixClusterAccelBuildInputGrids
- struct OptixClusterAccelBuildInputClusters
- struct OptixClusterAccelPrimitiveInfo
- struct OptixClusterAccelBuildInputTrianglesArgs
- struct OptixClusterAccelBuildInputGridsArgs
- struct OptixClusterAccelBuildInputTemplatesArgs
- struct OptixClusterAccelBuildInputClustersArgs
- struct OptixClusterAccelBuildInput
- struct OptixClusterAccelBuildModeDesc
- struct OptixImage2D
- struct OptixDenoiserOptions
- struct OptixDenoiserGuideLayer
- struct OptixDenoiserLayer
- struct OptixDenoiserParams
- struct OptixDenoiserSizes
- struct OptixTraverseData
- struct OptixModuleCompileBoundValueEntry
- struct OptixPayloadType
- struct OptixModuleCompileOptions
- struct OptixBuiltinISOptions
- struct OptixProgramGroupSingleModule
- struct OptixProgramGroupHitgroup
- struct OptixProgramGroupCallables

- struct OptixProgramGroupDesc
- struct OptixProgramGroupOptions
- struct OptixPipelineCompileOptions
- struct OptixPipelineLinkOptions
- struct OptixShaderBindingTable
- struct OptixStackSizes
- struct OptixCoopVecMatrixDescription
- struct OptixNetworkDescription

Macros

- #define OPTIX_SBT_RECORD_HEADER_SIZE ((size_t)32)
- #define OPTIX_SBT_RECORD_ALIGNMENT 16ull
- #define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128ull
- #define OPTIX_INSTANCE_BYTE_ALIGNMENT 16ull
- #define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8ull
- #define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16ull
- #define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
- #define OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
- #define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
- #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT (0)
- #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE (1)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT (2)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE (3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT (-1)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE (-2)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT (-3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE (-4)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_CLUSTER_SKIP_OPACITY_MICROMAP (-5)
- #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
- #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12

Typedefs

- typedef unsigned long long CUdeviceptr
- typedef struct OptixDeviceContext_t * OptixDeviceContext
- typedef struct OptixModule_t * OptixModule
- typedef struct OptixProgramGroup_t * OptixProgramGroup
- typedef struct OptixPipeline_t * OptixPipeline
- typedef struct OptixDenoiser_t * OptixDenoiser
- typedef struct OptixTask_t * OptixTask
- typedef unsigned long long OptixTraversableHandle
- typedef unsigned int OptixVisibilityMask
- typedef enum OptixResult OptixResult
- typedef enum OptixDeviceProperty OptixDeviceProperty
- typedef void(* OptixLogCallback) (unsigned int level, const char *tag, const char *message, void *cbdata)

- typedef enum OptixDeviceContextValidationMode OptixDeviceContextValidationMode
- typedef struct OptixDeviceContextOptions OptixDeviceContextOptions
- typedef enum OptixDevicePropertyShaderExecutionReorderingFlags OptixDevicePropertyShaderExecutionReorderingFlags
- typedef enum OptixDevicePropertyClusterAccelFlags OptixDevicePropertyClusterAccelFlags
- typedef enum OptixGeometryFlags OptixGeometryFlags
- typedef enum OptixHitKind OptixHitKind
- typedef enum OptixIndicesFormat OptixIndicesFormat
- typedef enum OptixVertexFormat OptixVertexFormat
- typedef enum OptixTransformFormat OptixTransformFormat
- typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat
- typedef enum OptixOpacityMicromapArrayIndexingMode OptixOpacityMicromapArrayIndexingMode
- typedef struct OptixOpacityMicromapUsageCount OptixOpacityMicromapUsageCount
- typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap
- typedef struct OptixRelocateInputOpacityMicromap OptixRelocateInputOpacityMicromap
- typedef struct OptixBuildInputTriangleArray OptixBuildInputTriangleArray
- typedef struct OptixRelocateInputTriangleArray OptixRelocateInputTriangleArray
- typedef enum OptixPrimitiveType OptixPrimitiveType
- typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags
- typedef enum OptixCurveEndcapFlags OptixCurveEndcapFlags
- typedef struct OptixBuildInputCurveArray OptixBuildInputCurveArray
- typedef struct OptixBuildInputSphereArray OptixBuildInputSphereArray
- typedef struct OptixAabb OptixAabb
- typedef struct OptixBuildInputCustomPrimitiveArray OptixBuildInputCustomPrimitiveArray
- typedef struct OptixBuildInputInstanceArray OptixBuildInputInstanceArray
- typedef struct OptixRelocateInputInstanceArray OptixRelocateInputInstanceArray
- typedef enum OptixBuildInputType OptixBuildInputType
- typedef struct OptixBuildInput OptixBuildInput
- typedef struct OptixRelocateInput OptixRelocateInput
- typedef enum OptixInstanceFlags OptixInstanceFlags
- typedef struct OptixInstance OptixInstance
- typedef enum OptixBuildFlags OptixBuildFlags
- typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags
- typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc
- typedef struct OptixOpacityMicromapHistogramEntry OptixOpacityMicromapHistogramEntry
- typedef struct OptixOpacityMicromapArrayBuildInput OptixOpacityMicromapArrayBuildInput
- typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
- typedef struct OptixMicromapBuffers OptixMicromapBuffers
- typedef enum OptixBuildOperation OptixBuildOperation
- typedef enum OptixMotionFlags OptixMotionFlags
- typedef struct OptixMotionOptions OptixMotionOptions
- typedef struct OptixAccelBuildOptions OptixAccelBuildOptions
- typedef struct OptixAccelBufferSizes OptixAccelBufferSizes
- typedef enum OptixAccelPropertyType OptixAccelPropertyType
- typedef struct OptixAccelEmitDesc OptixAccelEmitDesc
- typedef struct OptixRelocationInfo OptixRelocationInfo
- typedef struct OptixStaticTransform OptixStaticTransform
- typedef struct OptixMatrixMotionTransform OptixMatrixMotionTransform
- typedef struct OptixSRTData OptixSRTData

- typedef struct OptixSRTMotionTransform OptixSRTMotionTransform
- typedef enum OptixTraversableType OptixTraversableType
- typedef enum OptixClusterAccelBuildFlags OptixClusterAccelBuildFlags
- typedef enum OptixClusterAccelClusterFlags OptixClusterAccelClusterFlags
- typedef enum OptixClusterAccelPrimitiveFlags OptixClusterAccelPrimitiveFlags
- typedef enum OptixClusterAccelBuildType OptixClusterAccelBuildType
- typedef enum OptixClusterAccelBuildMode OptixClusterAccelBuildMode
- typedef enum OptixClusterAccelIndicesFormat OptixClusterAccelIndicesFormat
- typedef struct OptixClusterAccelBuildModeDescImplicitDest OptixClusterAccelBuildModeDescImplicitDest
- typedef struct OptixClusterAccelBuildModeDescExplicitDest OptixClusterAccelBuildModeDescExplicitDest
- typedef struct OptixClusterAccelBuildModeDescGetSize OptixClusterAccelBuildModeDescGetSize
- typedef struct OptixClusterAccelBuildInputTriangles OptixClusterAccelBuildInputTriangles
- typedef struct OptixClusterAccelBuildInputGrids OptixClusterAccelBuildInputGrids
- typedef struct OptixClusterAccelBuildInputClusters OptixClusterAccelBuildInputClusters
- typedef struct OptixClusterAccelPrimitiveInfo OptixClusterAccelPrimitiveInfo
- typedef enum OptixClusterIDValues OptixClusterIDValues
- typedef struct OptixClusterAccelBuildInputTrianglesArgs OptixClusterAccelBuildInputTrianglesArgs
- typedef struct OptixClusterAccelBuildInputGridsArgs OptixClusterAccelBuildInputGridsArgs
- typedef struct OptixClusterAccelBuildInputTemplatesArgs OptixClusterAccelBuildInputTemplatesArgs
- typedef struct OptixClusterAccelBuildInputClustersArgs OptixClusterAccelBuildInputClustersArgs
- typedef struct OptixClusterAccelBuildInput OptixClusterAccelBuildInput
- typedef struct OptixClusterAccelBuildModeDesc OptixClusterAccelBuildModeDesc
- typedef enum OptixPixelFormat OptixPixelFormat
- typedef struct OptixImage2D OptixImage2D
- typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
- typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
- typedef struct OptixDenoiserOptions OptixDenoiserOptions
- typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
- typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
- typedef struct OptixDenoiserLayer OptixDenoiserLayer
- typedef struct OptixDenoiserParams OptixDenoiserParams
- typedef struct OptixDenoiserSizes OptixDenoiserSizes
- typedef enum OptixRayFlags OptixRayFlags
- typedef enum OptixTransformType OptixTransformType
- typedef struct OptixTraverseData OptixTraverseData
- typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags
- typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
- typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
- typedef enum OptixModuleCompileState OptixModuleCompileState
- typedef struct OptixModuleCompileBoundValueEntry OptixModuleCompileBoundValueEntry
- typedef enum OptixPayloadTypeID OptixPayloadTypeID
- typedef enum OptixPayloadSemantics OptixPayloadSemantics
- typedef struct OptixPayloadType OptixPayloadType
- typedef struct OptixModuleCompileOptions OptixModuleCompileOptions

- typedef struct OptixBuiltinISOptions OptixBuiltinISOptions
- typedef enum OptixProgramGroupKind OptixProgramGroupKind
- typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
- typedef struct OptixProgramGroupSingleModule OptixProgramGroupSingleModule
- typedef struct OptixProgramGroupHitgroup OptixProgramGroupHitgroup
- typedef struct OptixProgramGroupCallables OptixProgramGroupCallables
- typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
- typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
- typedef enum OptixExceptionCodes OptixExceptionCodes
- typedef enum OptixExceptionFlags OptixExceptionFlags
- typedef struct OptixPipelineCompileOptions OptixPipelineCompileOptions
- typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
- typedef struct OptixShaderBindingTable OptixShaderBindingTable
- typedef struct OptixStackSizes OptixStackSizes
- typedef enum OptixDevicePropertyCoopVecFlags OptixDevicePropertyCoopVecFlags
- typedef enum OptixCoopVecElemType OptixCoopVecElemType
- typedef enum OptixCoopVecMatrixLayout OptixCoopVecMatrixLayout
- typedef struct OptixCoopVecMatrixDescription OptixCoopVecMatrixDescription
- typedef struct OptixNetworkDescription OptixNetworkDescription
- typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
- typedef OptixResult() OptixQueryFunctionTable_t(int abiId, unsigned int numOptions, OptixQueryFunctionTableOptions *, const void **, void *functionTable, size_t sizeOfTable)

Enumerations

- enum OptixResult {
  OPTIX_SUCCESS = 0 ,
  OPTIX_ERROR_INVALID_VALUE = 7001 ,
  OPTIX_ERROR_HOST_OUT_OF_MEMORY = 7002 ,
  OPTIX_ERROR_INVALID_OPERATION = 7003 ,
  OPTIX_ERROR_FILE_IO_ERROR = 7004 ,
  OPTIX_ERROR_INVALID_FILE_FORMAT = 7005 ,
  OPTIX_ERROR_DISK_CACHE_INVALID_PATH = 7010 ,
  OPTIX_ERROR_DISK_CACHE_PERMISSION_ERROR = 7011 ,
  OPTIX_ERROR_DISK_CACHE_DATABASE_ERROR = 7012 ,
  OPTIX_ERROR_DISK_CACHE_INVALID_DATA = 7013 ,
  OPTIX_ERROR_LAUNCH_FAILURE = 7050 ,
  OPTIX_ERROR_INVALID_DEVICE_CONTEXT = 7051 ,
  OPTIX_ERROR_CUDA_NOT_INITIALIZED = 7052 ,
  OPTIX_ERROR_VALIDATION_FAILURE = 7053 ,
  OPTIX_ERROR_INVALID_INPUT = 7200 ,
  OPTIX_ERROR_INVALID_LAUNCH_PARAMETER = 7201 ,
  OPTIX_ERROR_INVALID_PAYLOAD_ACCESS = 7202 ,
  OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS = 7203 ,
  OPTIX_ERROR_INVALID_FUNCTION_USE = 7204 ,
  OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS = 7205 ,
  OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250 ,
  OPTIX_ERROR_PIPELINE_LINK_ERROR = 7251 ,
  OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE = 7270 ,
  OPTIX_ERROR_INTERNAL_COMPILER_ERROR = 7299 ,
  OPTIX_ERROR_DENOISER_MODEL_NOT_SET = 7300 ,
  OPTIX_ERROR_DENOISER_NOT_INITIALIZED = 7301 ,

OPTIX_ERROR_NOT_COMPATIBLE = 7400 ,
OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH = 7500 ,
OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED = 7501 ,
OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID = 7502 ,
OPTIX_ERROR_NOT_SUPPORTED = 7800 ,
OPTIX_ERROR_UNSUPPORTED_ABI_VERSION = 7801 ,
OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH = 7802 ,
OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS = 7803 ,
OPTIX_ERROR_LIBRARY_NOT_FOUND = 7804 ,
OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND = 7805 ,
OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE = 7806 ,
OPTIX_ERROR_DEVICE_OUT_OF_MEMORY = 7807 ,
OPTIX_ERROR_INVALID_POINTER = 7808 ,
OPTIX_ERROR_CUDA_ERROR = 7900 ,
OPTIX_ERROR_INTERNAL_ERROR = 7990 ,
OPTIX_ERROR_UNKNOWN = 7999 }

- enum OptixDeviceProperty {
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004 ,
  OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009 ,
  OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING = 0x200A ,
  OPTIX_DEVICE_PROPERTY_COOP_VEC = 0x200B ,
  OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL = 0x2020 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_VERTICES = 0x2021 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_TRIANGLES = 0x2022 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_STRUCTURED_GRID_RESOLUTION = 0x2023 }
- 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 OptixDevicePropertyClusterAccelFlags {
  OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_NONE = 0 ,
  OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_STANDARD = 1 << 0 }
- enum OptixGeometryFlags {
  OPTIX_GEOMETRY_FLAG_NONE = 0 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
  OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u << 1 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 2 }
- enum OptixHitKind {
  OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE ,
  OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF }
- enum OptixIndicesFormat {
  OPTIX_INDICES_FORMAT_NONE = 0 ,
  OPTIX_INDICES_FORMAT_UNSIGNED_BYTE3 = 0x2101 ,

OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102 ,
OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103 }

- enum OptixVertexFormat {
OPTIX_VERTEX_FORMAT_NONE = 0 ,
OPTIX_VERTEX_FORMAT_FLOAT3 = 0x2121 ,
OPTIX_VERTEX_FORMAT_FLOAT2 = 0x2122 ,
OPTIX_VERTEX_FORMAT_HALF3 = 0x2123 ,
OPTIX_VERTEX_FORMAT_HALF2 = 0x2124 ,
OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125 ,
OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126 }
- enum OptixTransformFormat {
OPTIX_TRANSFORM_FORMAT_NONE = 0 ,
OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1 }
- enum OptixOpacityMicromapFormat {
OPTIX_OPACITY_MICROMAP_FORMAT_NONE = 0 ,
OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE = 1 ,
OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE = 2 }
- enum OptixOpacityMicromapArrayIndexingMode {
OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0 ,
OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1 ,
OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2 }
- enum OptixPrimitiveType {
OPTIX_PRIMITIVE_TYPE_CUSTOM = 0x2500 ,
OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE = 0x2501 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE = 0x2502 ,
OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR = 0x2503 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM = 0x2504 ,
OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE = 0x2505 ,
OPTIX_PRIMITIVE_TYPE_SPHERE = 0x2506 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER = 0x2507 ,
OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE_ROCAPS = 0x2508 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE_ROCAPS = 0x2509 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM_ROCAPS = 0x250A ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER_ROCAPS = 0x250B ,
OPTIX_PRIMITIVE_TYPE_TRIANGLE = 0x2531 }
- enum OptixPrimitiveTypeFlags {
OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM = 1 << 0 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE = 1 << 1 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE = 1 << 2 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR = 1 << 3 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM = 1 << 4 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE = 1 << 5 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE = 1 << 6 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER = 1 << 7 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE_ROCAPS = 1 << 8 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE_ROCAPS = 1 << 9 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM_ROCAPS = 1 << 10 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER_ROCAPS = 1 << 11 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE = 1 << 31 }
- enum OptixCurveEndcapFlags {
OPTIX_CURVE_ENDCAP_DEFAULT = 0 ,
OPTIX_CURVE_ENDCAP_ON = 1 << 0 }

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

- enum OptixClusterAccelPrimitiveFlags {
  OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_NONE = 0 ,
  OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1 << 0 ,
  OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1 << 1 ,
  OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_ANYHIT = 1 << 2 }
- enum OptixClusterAccelBuildType {
  OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS = 0x2545 ,
  OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES = 0x2546 ,
  OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES = 0x2547 ,
  OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES = 0x2548 ,
  OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS = 0x2549 }
- enum OptixClusterAccelBuildMode {
  OPTIX_CLUSTER_ACCEL_BUILD_MODE_IMPLICIT_DESTINATIONS = 0 ,
  OPTIX_CLUSTER_ACCEL_BUILD_MODE_EXPLICIT_DESTINATIONS = 1 ,
  OPTIX_CLUSTER_ACCEL_BUILD_MODE_GET_SIZES = 2 }
- enum OptixClusterAccelIndicesFormat {
  OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_8BIT = 1 ,
  OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_16BIT = 2 ,
  OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_32BIT = 4 }
- enum OptixClusterIDValues { OPTIX_CLUSTER_ID_INVALID = 0xFFFFFFFFu }
- enum OptixPixelFormat {
  OPTIX_PIXEL_FORMAT_HALF1 = 0x220a ,
  OPTIX_PIXEL_FORMAT_HALF2 = 0x2207 ,
  OPTIX_PIXEL_FORMAT_HALF3 = 0x2201 ,
  OPTIX_PIXEL_FORMAT_HALF4 = 0x2202 ,
  OPTIX_PIXEL_FORMAT_FLOAT1 = 0x220b ,
  OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208 ,
  OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203 ,
  OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204 ,
  OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205 ,
  OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206 ,
  OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209 }
- enum OptixDenoiserModelKind {
  OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326 ,
  OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328 ,
  OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322 ,
  OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325 }
- enum OptixDenoiserAlphaMode {
  OPTIX_DENOISER_ALPHA_MODE_COPY = 0 ,
  OPTIX_DENOISER_ALPHA_MODE_DENOISE = 1 }
- enum OptixDenoiserAOVType {
  OPTIX_DENOISER_AOV_TYPE_NONE = 0 ,
  OPTIX_DENOISER_AOV_TYPE_BEAUTY = 0x7000 ,
  OPTIX_DENOISER_AOV_TYPE_SPECULAR = 0x7001 ,
  OPTIX_DENOISER_AOV_TYPE_REFLECTION = 0x7002 ,
  OPTIX_DENOISER_AOV_TYPE_REFRACTION = 0x7003 ,
  OPTIX_DENOISER_AOV_TYPE_DIFFUSE = 0x7004 }
- enum OptixRayFlags {
  OPTIX_RAY_FLAG_NONE = 0u ,

OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u << 1 ,
OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u << 2 ,
OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u << 3 ,
OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u << 4 ,
OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u << 5 ,
OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u << 6 ,
OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u << 7 ,
OPTIX_RAY_FLAG_SKIP_TRIANGLES = 1u << 8 ,
OPTIX_RAY_FLAG_SKIP_AABBS = 1u << 9 ,
OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 10 }

- enum OptixTransformType {
  OPTIX_TRANSFORM_TYPE_NONE = 0 ,
  OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM = 1 ,
  OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM = 2 ,
  OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM = 3 ,
  OPTIX_TRANSFORM_TYPE_INSTANCE = 4 }
- enum OptixTraversableGraphFlags {
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u << 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u << 1 }
- enum OptixCompileOptimizationLevel {
  OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343 }
- enum OptixCompileDebugLevel {
  OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT = 0 ,
  OPTIX_COMPILE_DEBUG_LEVEL_NONE = 0x2350 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL = 0x2351 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353 ,
  OPTIX_COMPILE_DEBUG_LEVEL_FULL = 0x2352 }
- enum OptixModuleCompileState {
  OPTIX_MODULE_COMPILE_STATE_NOT_STARTED = 0x2360 ,
  OPTIX_MODULE_COMPILE_STATE_STARTED = 0x2361 ,
  OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE = 0x2362 ,
  OPTIX_MODULE_COMPILE_STATE_FAILED = 0x2363 ,
  OPTIX_MODULE_COMPILE_STATE_COMPLETED = 0x2364 }
- enum OptixPayloadTypeID {
  OPTIX_PAYLOAD_TYPE_DEFAULT = 0 ,
  OPTIX_PAYLOAD_TYPE_ID_0 = (1 << 0u) ,
  OPTIX_PAYLOAD_TYPE_ID_1 = (1 << 1u) ,
  OPTIX_PAYLOAD_TYPE_ID_2 = (1 << 2u) ,
  OPTIX_PAYLOAD_TYPE_ID_3 = (1 << 3u) ,
  OPTIX_PAYLOAD_TYPE_ID_4 = (1 << 4u) ,
  OPTIX_PAYLOAD_TYPE_ID_5 = (1 << 5u) ,
  OPTIX_PAYLOAD_TYPE_ID_6 = (1 << 6u) ,
  OPTIX_PAYLOAD_TYPE_ID_7 = (1 << 7u) }
- enum OptixPayloadSemantics {
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ = 1u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE = 2u << 0 ,

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

- enum OptixProgramGroupKind {
  OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421 ,
  OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422 ,
  OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423 ,
  OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424 ,
  OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425 }
- enum OptixProgramGroupFlags { OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0 }
- enum OptixExceptionCodes {
  OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1 ,
  OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED = -3 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE = -5 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT = -6 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT = -7 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE = -8 ,
  OPTIX_EXCEPTION_CODE_INVALID_RAY = -9 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH = -10 ,
  OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH = -11 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT = -12 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD = -13 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD = -14 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS = -15 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0 = -16 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_1 = -17 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2 = -18 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS = -32 ,
  OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH = -33 }
- enum OptixExceptionFlags {
  OPTIX_EXCEPTION_FLAG_NONE = 0 ,
  OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u << 0 ,
  OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u << 1 ,
  OPTIX_EXCEPTION_FLAG_USER = 1u << 2 ,
  OPTIX_EXCEPTION_FLAG_DEBUG = 1u << 3 }
- enum OptixDevicePropertyCoopVecFlags {
  OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_NONE = 0 ,
  OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_STANDARD = 1 << 0 }

- enum OptixCoopVecElemType {
  OPTIX_COOP_VEC_ELEM_TYPE_UNKNOWN = 0x2A00 ,
  OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16 = 0x2A01 ,
  OPTIX_COOP_VEC_ELEM_TYPE_FLOAT32 = 0x2A03 ,
  OPTIX_COOP_VEC_ELEM_TYPE_UINT8 = 0x2A04 ,
  OPTIX_COOP_VEC_ELEM_TYPE_INT8 = 0x2A05 ,
  OPTIX_COOP_VEC_ELEM_TYPE_UINT32 = 0x2A08 ,
  OPTIX_COOP_VEC_ELEM_TYPE_INT32 = 0x2A09 ,
  OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E4M3 = 0x2A0A ,
  OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E5M2 = 0x2A0B }
- enum OptixCoopVecMatrixLayout {
  OPTIX_COOP_VEC_MATRIX_LAYOUT_ROW_MAJOR = 0x2A40 ,
  OPTIX_COOP_VEC_MATRIX_LAYOUT_COLUMN_MAJOR = 0x2A41 ,
  OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL = 0x2A42 ,
  OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL = 0x2A43 }
- enum OptixQueryFunctionTableOptions { OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0 }

5.16.1 Detailed Description

OptiX Types.

5.16.2 Macro Definition Documentation

5.16.2.1 OPTIX_AABB_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8ull
```

Alignment requirement for OptixBuildInputCustomPrimitiveArray::aabbBuffers.

5.16.2.2 OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128ull
```

Alignment requirement for output and temporay buffers for acceleration structures.

5.16.2.3 OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
```

Maximum number of payload types allowed.

5.16.2.4 OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
```

Maximum number of payload values allowed.

5.16.2.5 OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT

```
#define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
```

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

5.16.2.6 OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT

```
#define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16ull
```

Alignment requirement for OptixBuildInputTriangleArray::preTransform.

5.16.2.7 OPTIX_INSTANCE_BYTE_ALIGNMENT

```
#define OPTIX_INSTANCE_BYTE_ALIGNMENT 16ull
```

Alignment requirement for OptixBuildInputInstanceArray::instances.

5.16.2.8 OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
```

Alignment requirement for opacity micromap array buffers.

5.16.2.9 OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT

```
#define OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
```

Alignment requirement for OptixOpacityMicromapArrayBuildInput::perMicromapDescBuffer.

5.16.2.10 OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL

```
#define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
```

Maximum subdivision level for opacity micromaps.

5.16.2.11 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_CLUSTER_SKIP_OPACITY_MICRO

```
#define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_CLUSTER_SKIP_OPACITY_
MICROMAP (-5)
```

Predefined index to indicate that no opacity micromap applies for a triangle. The opaque/non-opaque state is determined by the geometry flags, similar as for triangles in instances with the OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS flag set. This special index is only available for the opacity micromap index array supplied to OptixClusterAccelBuildInputTrianglesArgs. This special index does NOT require the cluster to be built with OPTIX_CLUSTER_ACCEL_CLUSTER_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS.

5.16.2.12 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE

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

5.16.2.13 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT

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

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

5.16.2.14 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE

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

5.16.2.15 OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPAREN

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

5.16.2.16 OPTIX_OPACITY_MICROMAP_STATE_OPAQUE

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

5.16.2.17 OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT

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

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

5.16.2.18 OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE

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

5.16.2.19 OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT

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

5.16.2.20 OPTIX_SBT_RECORD_ALIGNMENT

```
#define OPTIX_SBT_RECORD_ALIGNMENT 16ull
```

Alignment requirement for device pointers in OptixShaderBindingTable.

5.16.2.21 OPTIX_SBT_RECORD_HEADER_SIZE

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

Size of the SBT record headers.

5.16.2.22 OPTIX_TRANSFORM_BYTE_ALIGNMENT

```
#define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
```

Alignment requirement for OptixStaticTransform, OptixMatrixMotionTransform, OptixSRTMotionTransform.

5.16.3 Typedef Documentation

5.16.3.1 CUdeviceptr

```
typedef unsigned long long CUdeviceptr
```

CUDA device pointer.

5.16.3.2 OptixAabb

```
typedef struct OptixAabb OptixAabb
```

AABB inputs.

5.16.3.3 OptixAccelBufferSizes

```
typedef struct OptixAccelBufferSizes OptixAccelBufferSizes
```

Struct for querying builder allocation requirements.

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

See also optixAccelComputeMemoryUsage( )

5.16.3.4 OptixAccelBuildOptions

```
typedef struct OptixAccelBuildOptions OptixAccelBuildOptions
```

Build options for acceleration structures.

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( )

5.16.3.5 OptixAccelEmitDesc

`typedef struct OptixAccelEmitDesc OptixAccelEmitDesc`

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

See also optixAccelBuild( )

5.16.3.6 OptixAccelPropertyType

`typedef enum OptixAccelPropertyType OptixAccelPropertyType`

Properties which can be emitted during acceleration structure build.

See also OptixAccelEmitDesc::type.

5.16.3.7 OptixBuildFlags

`typedef enum OptixBuildFlags OptixBuildFlags`

Builder Options.

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

5.16.3.8 OptixBuildInput

`typedef struct OptixBuildInput OptixBuildInput`

Build inputs.

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

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( )

5.16.3.9 OptixBuildInputCurveArray

`typedef struct OptixBuildInputCurveArray OptixBuildInputCurveArray`

Curve inputs.

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

5.16.3.10 OptixBuildInputCustomPrimitiveArray

`typedef struct OptixBuildInputCustomPrimitiveArray`

`OptixBuildInputCustomPrimitiveArray`

Custom primitive inputs.

See also OptixBuildInput::customPrimitiveArray

5.16.3.11 OptixBuildInputInstanceArray

```
typedef struct OptixBuildInputInstanceArray OptixBuildInputInstanceArray
```

Instance and instance pointer inputs.

See also OptixBuildInput::instanceArray

5.16.3.12 OptixBuildInputOpacityMicromap

```
typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap
```

5.16.3.13 OptixBuildInputSphereArray

```
typedef struct OptixBuildInputSphereArray OptixBuildInputSphereArray
```

Sphere inputs.

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

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

See also OptixBuildInput::sphereArray

5.16.3.14 OptixBuildInputTriangleArray

```
typedef struct OptixBuildInputTriangleArray OptixBuildInputTriangleArray
```

Triangle inputs.

See also OptixBuildInput::triangleArray

5.16.3.15 OptixBuildInputType

```
typedef enum OptixBuildInputType OptixBuildInputType
```

Enum to distinguish the different build input types.

See also OptixBuildInput::type

5.16.3.16 OptixBuildOperation

```
typedef enum OptixBuildOperation OptixBuildOperation
```

Enum to specify the acceleration build operation.

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

Acceleration structure updates utilize the same acceleration structure, but with updated bounds.

Updates are typically much faster than builds, however, large perturbations can degrade the quality of the acceleration structure.

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( ), OptixAccelBuildOptions

5.16.3.17 OptixBuiltinISOptions

```
typedef struct OptixBuiltinISOptions OptixBuiltinISOptions
```

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

See also optixBuiltinISModuleGet( )

5.16.3.18 OptixClusterAccelBuildFlags

```
typedef enum OptixClusterAccelBuildFlags OptixClusterAccelBuildFlags
```

Flags affect all builds of a multi indirect cluster build.

5.16.3.19 OptixClusterAccelBuildInput

```
typedef struct OptixClusterAccelBuildInput OptixClusterAccelBuildInput
```

5.16.3.20 OptixClusterAccelBuildInputClusters

```
typedef struct OptixClusterAccelBuildInputClusters
OptixClusterAccelBuildInputClusters
```

5.16.3.21 OptixClusterAccelBuildInputClustersArgs

```
typedef struct OptixClusterAccelBuildInputClustersArgs
OptixClusterAccelBuildInputClustersArgs
```

Device data, args provided for OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS builds.

5.16.3.22 OptixClusterAccelBuildInputGrids

```
typedef struct OptixClusterAccelBuildInputGrids
OptixClusterAccelBuildInputGrids
```

5.16.3.23 OptixClusterAccelBuildInputGridsArgs

```
typedef struct OptixClusterAccelBuildInputGridsArgs
OptixClusterAccelBuildInputGridsArgs
```

Device data, args provided for OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS builds.

5.16.3.24 OptixClusterAccelBuildInputTemplatesArgs

```
typedef struct OptixClusterAccelBuildInputTemplatesArgs
OptixClusterAccelBuildInputTemplatesArgs
```

Device data, args provided for OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES builds.

5.16.3.25 OptixClusterAccelBuildInputTriangles

```
typedef struct OptixClusterAccelBuildInputTriangles
OptixClusterAccelBuildInputTriangles
```

5.16.3.26 OptixClusterAccelBuildInputTrianglesArgs

```
typedef struct OptixClusterAccelBuildInputTrianglesArgs
OptixClusterAccelBuildInputTrianglesArgs
```

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

5.16.3.27 OptixClusterAccelBuildMode

```
typedef enum OptixClusterAccelBuildMode OptixClusterAccelBuildMode
```

5.16.3.28 OptixClusterAccelBuildModeDesc

```
typedef struct OptixClusterAccelBuildModeDesc OptixClusterAccelBuildModeDesc
```

5.16.3.29 OptixClusterAccelBuildModeDescExplicitDest

```
typedef struct OptixClusterAccelBuildModeDescExplicitDest
OptixClusterAccelBuildModeDescExplicitDest
```

5.16.3.30 OptixClusterAccelBuildModeDescGetSize

```
typedef struct OptixClusterAccelBuildModeDescGetSize
OptixClusterAccelBuildModeDescGetSize
```

5.16.3.31 OptixClusterAccelBuildModeDescImplicitDest

```
typedef struct OptixClusterAccelBuildModeDescImplicitDest
OptixClusterAccelBuildModeDescImplicitDest
```

5.16.3.32 OptixClusterAccelBuildType

```
typedef enum OptixClusterAccelBuildType OptixClusterAccelBuildType
```

5.16.3.33 OptixClusterAccelClusterFlags

```
typedef enum OptixClusterAccelClusterFlags OptixClusterAccelClusterFlags
```

Flags for building CLAS.

5.16.3.34 OptixClusterAccelIndicesFormat

```
typedef enum OptixClusterAccelIndicesFormat OptixClusterAccelIndicesFormat
```

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

5.16.3.35 OptixClusterAccelPrimitiveFlags

```
typedef enum OptixClusterAccelPrimitiveFlags OptixClusterAccelPrimitiveFlags
```

5.16.3.36 OptixClusterAccelPrimitiveInfo

```
typedef struct OptixClusterAccelPrimitiveInfo OptixClusterAccelPrimitiveInfo
```

5.16.3.37 OptixClusterIDValues

```
typedef enum OptixClusterIDValues OptixClusterIDValues
```

Reserved value for cluster IDs in Args.

5.16.3.38 OptixCompileDebugLevel

```
typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
```

Debug levels.

See also OptixModuleCompileOptions::debugLevel

5.16.3.39 OptixCompileOptimizationLevel

```
typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
```

Optimization levels.

See also OptixModuleCompileOptions::optLevel

5.16.3.40 OptixCoopVecElemType

```
typedef enum OptixCoopVecElemType OptixCoopVecElemType
```

5.16.3.41 OptixCoopVecMatrixDescription

```
typedef struct OptixCoopVecMatrixDescription OptixCoopVecMatrixDescription
```

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

The rowColumnStrideInBytes is ignored if the layout is either OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL or OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL

5.16.3.42 OptixCoopVecMatrixLayout

```
typedef enum OptixCoopVecMatrixLayout OptixCoopVecMatrixLayout
```

5.16.3.43 OptixCurveEndcapFlags

```
typedef enum OptixCurveEndcapFlags OptixCurveEndcapFlags
```

Curve end cap types, for non-linear curves.

5.16.3.44 OptixDenoiser

```
typedef struct OptixDenoiser_t* OptixDenoiser
```

Opaque type representing a denoiser instance.

5.16.3.45 OptixDenoiserAlphaMode

```
typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
```

Alpha denoising mode.

See also optixDenoiserCreate( )

5.16.3.46 OptixDenoiserAOVType

```
typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
```

AOV type used by the denoiser.

5.16.3.47 OptixDenoiserGuideLayer

```
typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
```

Guide layer for the denoiser.

See also optixDenoiserInvoke( )

5.16.3.48 OptixDenoiserLayer

```
typedef struct OptixDenoiserLayer OptixDenoiserLayer
```

Input/Output layers for the denoiser.

See also optixDenoiserInvoke( )

5.16.3.49 OptixDenoiserModelKind

```
typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
```

Model kind used by the denoiser.

See also optixDenoiserCreate

5.16.3.50 OptixDenoiserOptions

```
typedef struct OptixDenoiserOptions OptixDenoiserOptions
```

Options used by the denoiser.

See also optixDenoiserCreate( )

5.16.3.51 OptixDenoiserParams

```
typedef struct OptixDenoiserParams OptixDenoiserParams
```

Various parameters used by the denoiser.

See also optixDenoiserInvoke( )

optixDenoiserComputeIntensity( )

optixDenoiserComputeAverageColor( )

5.16.3.52 OptixDenoiserSizes

```
typedef struct OptixDenoiserSizes OptixDenoiserSizes
```

Various sizes related to the denoiser.

See also optixDenoiserComputeMemoryResources( )

5.16.3.53 OptixDeviceContext

```
typedef struct OptixDeviceContext_t* OptixDeviceContext
```

Opaque type representing a device context.

5.16.3.54 OptixDeviceContextOptions

```
typedef struct OptixDeviceContextOptions OptixDeviceContextOptions
```

Parameters used for optixDeviceContextCreate( )

See also optixDeviceContextCreate( )

5.16.3.55 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.

Currently, when not OFF all builtin debug exceptions are enabled and each thrown builtin exception will hard-stop program execution at the end of the exception program run, both for the default or user-provided exception programs. If really needed this could be fine-tuned like eg

- OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_DEBUG_EXCEPTIONS
- ...

See also optixDeviceContextCreate( )

5.16.3.56 OptixDeviceProperty

```
typedef enum OptixDeviceProperty OptixDeviceProperty
```

Parameters used for optixDeviceContextGetProperty( )

See also optixDeviceContextGetProperty( )

5.16.3.57 OptixDevicePropertyClusterAccelFlags

```
typedef enum OptixDevicePropertyClusterAccelFlags
OptixDevicePropertyClusterAccelFlags
```

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

See also optixDeviceContextGetProperty( )

5.16.3.58 OptixDevicePropertyCoopVecFlags

```
typedef enum OptixDevicePropertyCoopVecFlags OptixDevicePropertyCoopVecFlags
```

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

See also optixDeviceContextGetProperty( )

5.16.3.59 OptixDevicePropertyShaderExecutionReorderingFlags

```
typedef enum OptixDevicePropertyShaderExecutionReorderingFlags
OptixDevicePropertyShaderExecutionReorderingFlags
```

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

See also optixDeviceContextGetProperty( )

5.16.3.60 OptixExceptionCodes

`typedef enum OptixExceptionCodes OptixExceptionCodes`

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

These flags are interpreted on the device by rtcore, and should mirror RtcExceptionCodes.

5.16.3.61 OptixExceptionFlags

`typedef enum OptixExceptionFlags OptixExceptionFlags`

Exception flags.

See also OptixPipelineCompileOptions::exceptionFlags, OptixExceptionCodes These flags are interpreted on the device by rtcore, and should mirror RtcExceptionFlags.

5.16.3.62 OptixGeometryFlags

`typedef enum OptixGeometryFlags OptixGeometryFlags`

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

5.16.3.63 OptixHitKind

`typedef enum OptixHitKind OptixHitKind`

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

See also optixGetHitKind( )

5.16.3.64 OptixImage2D

`typedef struct OptixImage2D OptixImage2D`

Image descriptor used by the denoiser.

See also optixDenoiserInvoke( ), optixDenoiserComputeIntensity( )

5.16.3.65 OptixIndicesFormat

`typedef enum OptixIndicesFormat OptixIndicesFormat`

Format of indices used int OptixBuildInputTriangleArray::indexFormat.

5.16.3.66 OptixInstance

`typedef struct OptixInstance OptixInstance`

Instances.

See also OptixBuildInputInstanceArray::instances This struct is interpreted on the device by rtcore, and should mirror the RtcFatInstance.

5.16.3.67 OptixInstanceFlags

`typedef enum OptixInstanceFlags OptixInstanceFlags`

Flags set on the OptixInstance::flags.

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

These flags are interpreted on the device by rtcore, and should mirror the RtcInstanceFlags.

5.16.3.68 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

5.16.3.69 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( ) This struct is interpreted on the device by rtcore, and should mirror RtcTravMatrixMotionTransform.

5.16.3.70 OptixMicromapBuffers

```
typedef struct OptixMicromapBuffers OptixMicromapBuffers
```

Buffer inputs for opacity micromap array builds.

5.16.3.71 OptixMicromapBufferSizes

```
typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
```

Conservative memory requirements for building a opacity micromap array.

5.16.3.72 OptixModule

```
typedef struct OptixModule_t* OptixModule
```

Opaque type representing a module.

5.16.3.73 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

5.16.3.74 OptixModuleCompileOptions

```
typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
```

Compilation options for module.

See also optixModuleCreate( )

5.16.3.75 OptixModuleCompileState

`typedef enum OptixModuleCompileState OptixModuleCompileState`

Module compilation state.

See also optixModuleGetCompilationState( ), optixModuleCreateWithTasks( )

5.16.3.76 OptixMotionFlags

`typedef enum OptixMotionFlags OptixMotionFlags`

Enum to specify motion flags.

See also OptixMotionOptions::flags.

5.16.3.77 OptixMotionOptions

`typedef struct OptixMotionOptions OptixMotionOptions`

Motion options.

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

5.16.3.78 OptixNetworkDescription

`typedef struct OptixNetworkDescription OptixNetworkDescription`

5.16.3.79 OptixOpacityMicromapArrayBuildInput

`typedef struct OptixOpacityMicromapArrayBuildInput OptixOpacityMicromapArrayBuildInput`

Inputs to opacity micromap array construction.

5.16.3.80 OptixOpacityMicromapArrayIndexingMode

`typedef enum OptixOpacityMicromapArrayIndexingMode OptixOpacityMicromapArrayIndexingMode`

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

5.16.3.81 OptixOpacityMicromapDesc

`typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc`

Opacity micromap descriptor.

5.16.3.82 OptixOpacityMicromapFlags

`typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags`

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

5.16.3.83 OptixOpacityMicromapFormat

`typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat`

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

5.16.3.84 OptixOpacityMicromapHistogramEntry

```
typedef struct OptixOpacityMicromapHistogramEntry
OptixOpacityMicromapHistogramEntry
```

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

5.16.3.85 OptixOpacityMicromapUsageCount

```
typedef struct OptixOpacityMicromapUsageCount OptixOpacityMicromapUsageCount
```

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

5.16.3.86 OptixPayloadSemantics

```
typedef enum OptixPayloadSemantics OptixPayloadSemantics
```

Semantic flags for a single payload word.

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

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

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

5.16.3.87 OptixPayloadType

```
typedef struct OptixPayloadType OptixPayloadType
```

Specifies a single payload type.

5.16.3.88 OptixPayloadTypeID

```
typedef enum OptixPayloadTypeID OptixPayloadTypeID
```

Payload type identifiers.

5.16.3.89 OptixPipeline

```
typedef struct OptixPipeline_t* OptixPipeline
```

Opaque type representing a pipeline.

5.16.3.90 OptixPipelineCompileOptions

```
typedef struct OptixPipelineCompileOptions OptixPipelineCompileOptions
```

Compilation options for all modules of a pipeline.

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

pipeline.

See also optixModuleCreate( ), optixPipelineCreate( )

5.16.3.91 OptixPipelineLinkOptions

`typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions`

Link options for a pipeline.

See also optixPipelineCreate( )

5.16.3.92 OptixPixelFormat

`typedef enum OptixPixelFormat OptixPixelFormat`

Pixel formats used by the denoiser.

See also OptixImage2D::format

5.16.3.93 OptixPrimitiveType

`typedef enum OptixPrimitiveType OptixPrimitiveType`

Builtin primitive types.

5.16.3.94 OptixPrimitiveTypeFlags

`typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags`

Builtin flags may be bitwise combined.

See also OptixPipelineCompileOptions::usesPrimitiveTypeFlags

5.16.3.95 OptixProgramGroup

`typedef struct OptixProgramGroup_t* OptixProgramGroup`

Opaque type representing a program group.

5.16.3.96 OptixProgramGroupCallables

`typedef struct OptixProgramGroupCallables OptixProgramGroupCallables`

Program group representing callables.

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

See also #OptixProgramGroupDesc::callables

5.16.3.97 OptixProgramGroupDesc

`typedef struct OptixProgramGroupDesc OptixProgramGroupDesc`

Descriptor for program groups.

5.16.3.98 OptixProgramGroupFlags

`typedef enum OptixProgramGroupFlags OptixProgramGroupFlags`

Flags for program groups.

5.16.3.99 OptixProgramGroupHitgroup

`typedef struct OptixProgramGroupHitgroup OptixProgramGroupHitgroup`

Program group representing the hitgroup.

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

See also OptixProgramGroupDesc::hitgroup

5.16.3.100 OptixProgramGroupKind

`typedef enum OptixProgramGroupKind OptixProgramGroupKind`

Distinguishes different kinds of program groups.

5.16.3.101 OptixProgramGroupOptions

`typedef struct OptixProgramGroupOptions OptixProgramGroupOptions`

Program group options.

See also optixProgramGroupCreate( )

5.16.3.102 OptixProgramGroupSingleModule

`typedef struct OptixProgramGroupSingleModule OptixProgramGroupSingleModule`

Program group representing a single module.

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

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

5.16.3.103 OptixQueryFunctionTable_t

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

Type of the function `optixQueryFunctionTable( )`

5.16.3.104 OptixQueryFunctionTableOptions

`typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions`

Options that can be passed to `optixQueryFunctionTable( )`

5.16.3.105 OptixRayFlags

`typedef enum OptixRayFlags OptixRayFlags`

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

See also optixTrace( ) These flags are interpreted on the device by rtcore, and should mirror RtcRayFlags.

5.16.3.106 OptixRelocateInput

```
typedef struct OptixRelocateInput OptixRelocateInput
```

Relocation inputs.

See also optixAccelRelocate( )

5.16.3.107 OptixRelocateInputInstanceArray

```
typedef struct OptixRelocateInputInstanceArray
OptixRelocateInputInstanceArray
```

Instance and instance pointer inputs.

See also OptixRelocateInput::instanceArray

5.16.3.108 OptixRelocateInputOpacityMicromap

```
typedef struct OptixRelocateInputOpacityMicromap
OptixRelocateInputOpacityMicromap
```

5.16.3.109 OptixRelocateInputTriangleArray

```
typedef struct OptixRelocateInputTriangleArray
OptixRelocateInputTriangleArray
```

Triangle inputs.

See also OptixRelocateInput::triangleArray

5.16.3.110 OptixRelocationInfo

```
typedef struct OptixRelocationInfo OptixRelocationInfo
```

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

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

5.16.3.111 OptixResult

```
typedef enum OptixResult OptixResult
```

Result codes returned from API functions.

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

See also optixGetErrorName( ), optixGetErrorString( )

5.16.3.112 OptixShaderBindingTable

```
typedef struct OptixShaderBindingTable OptixShaderBindingTable
```

Describes the shader binding table (SBT)

See also optixLaunch( )

5.16.3.113 OptixSRTData

```
typedef struct OptixSRTData OptixSRTData
```

Represents an SRT transformation.

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

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

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

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

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

See also OptixSRTMotionTransform::srtData, optixConvertPointerToTraversableHandle( )

5.16.3.114 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( ) This struct is interpreted on the device by rtcore, and should mirror RtcTravSRTMotionTransform.

5.16.3.115 OptixStackSizes

`typedef struct OptixStackSizes OptixStackSizes`

Describes the stack size requirements of a program group.

See also optixProgramGroupGetStackSize( )

5.16.3.116 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() This struct is interpreted on the device by rtcore, and should mirror RtcTravStaticTransform.

5.16.3.117 OptixTask

`typedef struct OptixTask_t* OptixTask`

Opaque type representing a work task.

5.16.3.118 OptixTransformFormat

`typedef enum OptixTransformFormat OptixTransformFormat`

Format of transform used in OptixBuildInputTriangleArray::transformFormat.

5.16.3.119 OptixTransformType

`typedef enum OptixTransformType OptixTransformType`

Transform.

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

The values of this enum are used on the device by rtcore, and should mirror RtcTransformType.

5.16.3.120 OptixTraversableGraphFlags

`typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags`

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

5.16.3.121 OptixTraversableHandle

`typedef unsigned long long OptixTraversableHandle`

Traversable handle.

5.16.3.122 OptixTraversableType

`typedef enum OptixTraversableType OptixTraversableType`

Traversable Handles.

See also optixConvertPointerToTraversableHandle()

5.16.3.123 OptixTraverseData

`typedef struct OptixTraverseData OptixTraverseData`

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

5.16.3.124 OptixVertexFormat

`typedef enum OptixVertexFormat OptixVertexFormat`

Format of vertices used in OptixBuildInputTriangleArray::vertexFormat.

5.16.3.125 OptixVisibilityMask

`typedef unsigned int OptixVisibilityMask`

Visibility mask.

5.16.4 Enumeration Type Documentation

5.16.4.1 OptixAccelPropertyType

`enum OptixAccelPropertyType`

Properties which can be emitted during acceleration structure build.

See also OptixAccelEmitDesc::type.

Enumerator

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

5.16.4.2 OptixBuildFlags

`enum OptixBuildFlags`

Builder Options.

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

Enumerator

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

Enumerator

OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS	Allow random access to build input vertices See optixGetTriangleVertexDataFromHandle optixGetLinearCurveVertexDataFromHandle optixGetQuadraticBSplineVertexDataFromHandle optixGetCubicBSplineVertexDataFromHandle optixGetCatmullRomVertexDataFromHandle optixGetCubicBezierVertexDataFromHandle optixGetQuadraticBSplineRocapsVertexDataFromHandle optixGetCubicBSplineRocapsVertexDataFromHandle optixGetCatmullRomRocapsVertexDataFromHandle optixGetCubicBezierRocapsVertexDataFromHandle optixGetRibbonVertexDataFromHandle optixGetRibbonNormalFromHandle optixGetSphereDataFromHandle.
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.

5.16.4.3 OptixBuildInputType

`enum OptixBuildInputType`

Enum to distinguish the different build input types.

See also OptixBuildInput::type

Enumerator

OPTIX_BUILD_INPUT_TYPE_TRIANGLES	Triangle inputs. See also OptixBuildInputTriangleArray
OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES	Custom primitive inputs. See also OptixBuildInputCustomPrimitiveArray

Enumerator

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

5.16.4.4 OptixBuildOperation

`enum OptixBuildOperation`

Enum to specify the acceleration build operation.

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

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

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( ), OptixAccelBuildOptions

Enumerator

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

5.16.4.5 OptixClusterAccelBuildFlags

`enum OptixClusterAccelBuildFlags`

Flags affect all builds of a multi indirect cluster build.

Enumerator

OPTIX_CLUSTER_ACCEL_BUILD_FLAG_NONE	
OPTIX_CLUSTER_ACCEL_BUILD_FLAG_PREFER_FAST_TRACE	
OPTIX_CLUSTER_ACCEL_BUILD_FLAG_PREFER_FAST_BUILD	
OPTIX_CLUSTER_ACCEL_BUILD_FLAG_ALLOW_OPACITY_MICROMAPS	

5.16.4.6 OptixClusterAccelBuildMode

`enum OptixClusterAccelBuildMode`

Enumerator

OPTIX_CLUSTER_ACCEL_BUILD_MODE_IMPLICIT_DESTINATIONS

Enumerator

OPTIX_CLUSTER_ACCEL_BUILD_MODE_EXPLICIT_DESTINATIONS
OPTIX_CLUSTER_ACCEL_BUILD_MODE_GET_SIZES

5.16.4.7 OptixClusterAccelBuildType

`enum OptixClusterAccelBuildType`

Enumerator

OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS
OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES
OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES
OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES
OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS

5.16.4.8 OptixClusterAccelClusterFlags

`enum OptixClusterAccelClusterFlags`

Flags for building CLAS.

Enumerator

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

5.16.4.9 OptixClusterAccelIndicesFormat

`enum OptixClusterAccelIndicesFormat`

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

Enumerator

OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_8BIT
OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_16BIT
OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_32BIT

5.16.4.10 OptixClusterAccelPrimitiveFlags

`enum OptixClusterAccelPrimitiveFlags`

Enumerator

OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_NONE
OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_TRIANGLE_FACE_CULLING
OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_REQUIRE_SINGLE_ANYHIT_CALL
OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_ANYHIT

5.16.4.11 OptixClusterIDValues

`enum OptixClusterIDValues`

Reserved value for cluster IDs in Args.

Enumerator

OPTIX_CLUSTER_ID_INVALID

5.16.4.12 OptixCompileDebugLevel

`enum OptixCompileDebugLevel`

Debug levels.

See also OptixModuleCompileOptions::debugLevel

Enumerator

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

5.16.4.13 OptixCompileOptimizationLevel

`enum OptixCompileOptimizationLevel`

Optimization levels.

See also OptixModuleCompileOptions::optLevel

Enumerator

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

5.16.4.14 OptixCoopVecElemType

`enum OptixCoopVecElemType`

Enumerator

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

5.16.4.15 OptixCoopVecMatrixLayout

`enum OptixCoopVecMatrixLayout`

Enumerator

OPTIX_COOP_VEC_MATRIX_LAYOUT_ROW_MAJOR
OPTIX_COOP_VEC_MATRIX_LAYOUT_COLUMN_MAJOR
OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL
OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL

5.16.4.16 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.

5.16.4.17 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.

5.16.4.18 OptixDenoiserAOVType

`enum OptixDenoiserAOVType`

AOV type used by the denoiser.

Enumerator

OPTIX_DENOISER_AOV_TYPE_NONE	Unspecified AOV type.
OPTIX_DENOISER_AOV_TYPE_BEAUTY	
OPTIX_DENOISER_AOV_TYPE_SPECULAR	
OPTIX_DENOISER_AOV_TYPE_REFLECTION	
OPTIX_DENOISER_AOV_TYPE_REFRACTION	
OPTIX_DENOISER_AOV_TYPE_DIFFUSE	

5.16.4.19 OptixDenoiserModelKind

`enum OptixDenoiserModelKind`

Model kind used by the denoiser.

See also optixDenoiserCreate

Enumerator

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

5.16.4.20 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.

Currently, when not OFF all builtin debug exceptions are enabled and each thrown builtin exception will hard-stop program execution at the end of the exception program run, both for the default or user-provided exception programs. If really needed this could be fine-tuned like eg

- OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_DEBUG_EXCEPTIONS
- ...

See also optixDeviceContextCreate( )

Enumerator

OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF
OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL

5.16.4.21 OptixDeviceProperty

`enum OptixDeviceProperty`

Parameters used for optixDeviceContextGetProperty( )

See also optixDeviceContextGetProperty( )

Enumerator

OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH	Maximum value for OptixPipelineLinkOptions ::maxTraceDepth. sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH	Maximum value to pass into optixPipelineSetStackSize for parameter maxTraversableGraphDepth. sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS	The maximum number of primitives (over all build inputs) as input to a single Geometry Acceleration Structure (GAS). sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS	The maximum number of instances (over all build inputs) as input to a single Instance Acceleration Structure (IAS). sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_RTCORE_VERSION	The RT core version supported by the device (0 for no support, 10 for version 1.0). sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID	The maximum value for OptixInstance ::instanceId. sizeof(unsigned int)

Enumerator

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)
OPTIX_DEVICE_PROPERTY_COOP_VEC	Returns a flag specifying whether cooperative vector support is enabled for this device. See OptixDevicePropertyCoopVecFlags for documentation on the values that can be returned. sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL	Returns a flag specifying support for cluster acceleration structure builds. See OptixDevicePropertyClusterAccelFlags for documentation on the values that can be returned. sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_VERTICES	Returns a maximum unique vertices per cluster in a cluster acceleration structure builds. sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_TRIANGLES	Returns a maximum triangles per cluster in a cluster acceleration structure builds. sizeof(unsigned int)
OPTIX_DEVICE_PROPERTY_LIMIT_MAX_STRUCTURED_GRID_RESOLUTION	Returns a maximum resolution per cluster in a structured cluster acceleration structure builds. sizeof(unsigned int)

5.16.4.22 OptixDevicePropertyClusterAccelFlags

`enum OptixDevicePropertyClusterAccelFlags`

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

See also optixDeviceContextGetProperty( )

Enumerator

OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_NONE	Cluster acceleration structure builds are not supported.

Enumerator

OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_STANDARD	

5.16.4.23 OptixDevicePropertyCoopVecFlags

`enum OptixDevicePropertyCoopVecFlags`

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

See also optixDeviceContextGetProperty( )

Enumerator

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

5.16.4.24 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.
OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD	

5.16.4.25 OptixExceptionCodes

`enum OptixExceptionCodes`

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

These flags are interpreted on the device by rtcore, and should mirror RtcExceptionCodes.

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.

Enumerator

OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED	The traversal depth is exceeded. Exception details: optixGetTransformListSize( ) optixGetTransformListHandle( )
OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE	Traversal encountered an invalid traversable type. Exception details: optixGetTransformListSize( ) optixGetTransformListHandle( ) optixGetExceptionInvalidTraversable( )
OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT	The miss SBT record index is out of bounds A miss SBT record index is valid within the range [0, OptixShaderBindingTable ::missRecordCount) (See optixLaunch) Exception details: optixGetExceptionInvalidSbtOffset( )
OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT	The traversal hit SBT record index out of bounds. A traversal hit SBT record index is valid within the range [0, OptixShaderBindingTable ::hitgroupRecordCount) (See optixLaunch) The following formula relates the sbt-geometry-acceleration-structure-index (See optixGetSbtGASIndex), sbt-stride-from-trace-call and sbt-offset-from-trace-call (See optixTrace) sbt-index = sbt-instance-offset + (sbt-geometry-acceleration-structure-index ∗ sbt-stride-from-trace-call) + sbt-offset-from-trace-call Exception details: optixGetTransformListSize( ) optixGetTransformListHandle( ) optixGetExceptionInvalidSbtOffset( ) optixGetSbtGASIndex( )
OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE	The shader encountered an unsupported primitive type (See OptixPipelineCompileOptions ::usesPrimitiveTypeFlags). no exception details.
OPTIX_EXCEPTION_CODE_INVALID_RAY	The shader encountered a call to optixTrace with at least one of the float arguments being inf or nan, or the tmin argument is negative. Exception details: optixGetExceptionInvalidRay( )
OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH	The shader encountered a call to either optixDirectCall or optixCallableCall where the argument count does not match the parameter count of the callable program which is called. Exception details: optixGetExceptionParameterMismatch.
OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH	The invoked builtin IS does not match the current GAS.

Enumerator

OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT	Tried to call a callable program using an SBT offset that is larger than the number of passed in callable SBT records. Exception details: optixGetExceptionInvalidSbtOffset( )
OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD	Tried to call a direct callable using an SBT offset of a record that was built from a program group that did not include a direct callable.
OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD	Tried to call a continuation callable using an SBT offset of a record that was built from a program group that did not include a continuation callable.
OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS	Tried to directly traverse a single gas while single gas traversable graphs are not enabled (see OptixTraversableGraphFlags::OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS). Exception details: optixGetTransformListSize( ) optixGetTransformListHandle( ) optixGetExceptionInvalidTraversable( )
OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0	argument passed to an optix call is not within an acceptable range of values.
OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_1	
OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2	
OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS	... reserved up to -31 Tried to access data on an AS without random data access support (See OptixBuildFlags).
OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH	The program payload type doesn't match the trace payload type.

5.16.4.26 OptixExceptionFlags

`enum OptixExceptionFlags`

Exception flags.

See also OptixPipelineCompileOptions::exceptionFlags, OptixExceptionCodes These flags are interpreted on the device by rtcore, and should mirror RtcExceptionFlags.

Enumerator

OPTIX_EXCEPTION_FLAG_NONE	No exception are enabled.

Enumerator

OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW	Enables exceptions check related to the continuation stack. This flag should be used when the application handles stack overflows in a user exception program as part of the normal flow of execution. For catching overflows during debugging and development, the device context validation mode should be used instead. See also 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( ).
OPTIX_EXCEPTION_FLAG_DEBUG	Enables various exceptions check related to traversal.

5.16.4.27 OptixGeometryFlags

`enum OptixGeometryFlags`

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

Enumerator

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

5.16.4.28 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.

5.16.4.29 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_BYTE3	Three bytes.
OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3	Three shorts.
OPTIX_INDICES_FORMAT_UNSIGNED_INT3	Three ints.

5.16.4.30 OptixInstanceFlags

`enum OptixInstanceFlags`

Flags set on the OptixInstance::flags.

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

These flags are interpreted on the device by rtcore, and should mirror the RtcInstanceFlags.

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.

Enumerator

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	Deprecated flag to disable the instance transformation. This flag never actually worked. If there where users they must have been setting the instance transform to identity, so the broken flag didn't cause any visible issues. WARNING: take care that any new flag overlapping with this deprecated flag won't trigger bugs in user code using an older SDK! OPTIX_INSTANCE_ FLAG_DISABLE_TRANSFORM = 1u << 6,. Force 4-state opacity micromaps to behave as 2-state opacity micromaps during traversal.
OPTIX_INSTANCE_FLAG_DISABLE_ OPACITY_MICROMAPS	Don't perform opacity micromap query for this instance. Triangle GAS must be built with ALLOW_DISABLE_OPACITY_MICROMAPS for this to be valid. Clusters in a GAS must be build with OPTIX_CLUSTER_ACCEL_ CLUSTER_FLAG_ALLOW_DISABLE_ OPACITY_MICROMAPS for this to be valid. This flag overrides FORCE_OPACTIY_ MIXROMAP_2_STATE instance and ray flags.

5.16.4.31 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.

Enumerator

OPTIX_MODULE_COMPILE_STATE_COMPLETED	All OptixTask objects have completed. The OptixModule is ready to be used.

5.16.4.32 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

5.16.4.33 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.

5.16.4.34 OptixOpacityMicromapFlags

`enum OptixOpacityMicromapFlags`

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

Enumerator

OPTIX_OPACITY_MICROMAP_FLAG_NONE	

Enumerator

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

5.16.4.35 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

5.16.4.36 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

Enumerator

OPTIX_PAYLOAD_SEMANTICS_MS_READ
OPTIX_PAYLOAD_SEMANTICS_MS_WRITE
OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE
OPTIX_PAYLOAD_SEMANTICS_AH_NONE
OPTIX_PAYLOAD_SEMANTICS_AH_READ
OPTIX_PAYLOAD_SEMANTICS_AH_WRITE
OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE
OPTIX_PAYLOAD_SEMANTICS_IS_NONE
OPTIX_PAYLOAD_SEMANTICS_IS_READ
OPTIX_PAYLOAD_SEMANTICS_IS_WRITE
OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE

5.16.4.37 OptixPayloadTypeID

`enum OptixPayloadTypeID`

Payload type identifiers.

Enumerator

OPTIX_PAYLOAD_TYPE_DEFAULT	
OPTIX_PAYLOAD_TYPE_ID_0	
OPTIX_PAYLOAD_TYPE_ID_1	
OPTIX_PAYLOAD_TYPE_ID_2	
OPTIX_PAYLOAD_TYPE_ID_3	
OPTIX_PAYLOAD_TYPE_ID_4	
OPTIX_PAYLOAD_TYPE_ID_5	
OPTIX_PAYLOAD_TYPE_ID_6	
OPTIX_PAYLOAD_TYPE_ID_7	

5.16.4.38 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

Enumerator

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

5.16.4.39 OptixPrimitiveType

`enum OptixPrimitiveType`

Builtin primitive types.

Enumerator

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

5.16.4.40 OptixPrimitiveTypeFlags

`enum OptixPrimitiveTypeFlags`

Builtin flags may be bitwise combined.

See also OptixPipelineCompileOptions::usesPrimitiveTypeFlags

Enumerator

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

5.16.4.41 OptixProgramGroupFlags

`enum OptixProgramGroupFlags`

Flags for program groups.

Enumerator

OPTIX_PROGRAM_GROUP_FLAGS_NONE	Currently there are no flags.

5.16.4.42 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

Enumerator

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

5.16.4.43 OptixQueryFunctionTableOptions

`enum OptixQueryFunctionTableOptions`

Options that can be passed to `optixQueryFunctionTable()`

Enumerator

OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY	Placeholder (there are no options yet)

5.16.4.44 OptixRayFlags

`enum OptixRayFlags`

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

See also optixTrace() These flags are interpreted on the device by rtcore, and should mirror RtcRayFlags.

Enumerator

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

Enumerator

OPTIX_RAY_FLAG_ENFORCE_ANYHIT	Forces anyhit program execution for the ray. Overrides OPTIX_GEOMETRY_FLAG_ DISABLE_ANYHIT as well as OPTIX_ INSTANCE_FLAG_DISABLE_ANYHIT. This flag is mutually exclusive with OPTIX_RAY_ FLAG_DISABLE_ANYHIT, OPTIX_RAY_FLAG _CULL_DISABLED_ANYHIT, OPTIX_RAY_ FLAG_CULL_ENFORCED_ANYHIT.
OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_ HIT	Terminates the ray after the first hit and executes the closesthit program of that hit.
OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT	Disables closesthit programs for the ray, but still executes miss program in case of a miss.
OPTIX_RAY_FLAG_CULL_BACK_FACING_ TRIANGLES	Do not intersect triangle back faces (respects a possible face change due to instance flag OPTIX _INSTANCE_FLAG_FLIP_TRIANGLE_ FACING). This flag is mutually exclusive with OPTIX_RAY_FLAG_CULL_FRONT_FACING_ TRIANGLES.
OPTIX_RAY_FLAG_CULL_FRONT_FACING_ TRIANGLES	Do not intersect triangle front faces (respects a possible face change due to instance flag OPTIX _INSTANCE_FLAG_FLIP_TRIANGLE_ FACING). This flag is mutually exclusive with OPTIX_RAY_FLAG_CULL_BACK_FACING_ TRIANGLES.
OPTIX_RAY_FLAG_CULL_DISABLED_ ANYHIT	Do not intersect geometry which disables anyhit programs (due to setting geometry flag OPTIX_ GEOMETRY_FLAG_DISABLE_ANYHIT or instance flag OPTIX_INSTANCE_FLAG_ DISABLE_ANYHIT). This flag is mutually exclusive with OPTIX_RAY_FLAG_CULL_ ENFORCED_ANYHIT, OPTIX_RAY_FLAG_ ENFORCE_ANYHIT, OPTIX_RAY_FLAG_ DISABLE_ANYHIT.
OPTIX_RAY_FLAG_CULL_ENFORCED_ ANYHIT	Do not intersect geometry which have an enabled anyhit program (due to not setting geometry flag OPTIX_GEOMETRY_FLAG_ DISABLE_ANYHIT or setting instance flag OPTIX_INSTANCE_FLAG_ENFORCE_ ANYHIT). This flag is mutually exclusive with OPTIX_RAY_FLAG_CULL_DISABLED_ ANYHIT, OPTIX_RAY_FLAG_ENFORCE_ ANYHIT, OPTIX_RAY_FLAG_DISABLE_ ANYHIT.
OPTIX_RAY_FLAG_SKIP_TRIANGLES	These flags are available in rtcore, but not exposed in optix. They are explicitly filtered out in lowerTrace. mutually exclusive with RTC_ RAY_FLAG_SKIP_AABBS.
OPTIX_RAY_FLAG_SKIP_AABBS	mutually exclusive with RTC_RAY_FLAG_SKIP _TRIANGLES.

Enumerator

OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE	Force 4-state opacity micromaps to behave as 2-state opacity micromaps during traversal.

5.16.4.45 OptixResult

`enum OptixResult`

Result codes returned from API functions.

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

See also optixGetErrorName( ), optixGetErrorString( )

Enumerator

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

Enumerator

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_INVALID_POINTER
OPTIX_ERROR_CUDA_ERROR
OPTIX_ERROR_INTERNAL_ERROR
OPTIX_ERROR_UNKNOWN

5.16.4.46 OptixTransformFormat

`enum OptixTransformFormat`

Format of transform used in OptixBuildInputTriangleArray::transformFormat.

Enumerator

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

5.16.4.47 OptixTransformType

`enum OptixTransformType`

Transform.

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

The values of this enum are used on the device by rtcore, and should mirror RtcTransformType.

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

5.16.4.48 OptixTraversableGraphFlags

`enum OptixTraversableGraphFlags`

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

Enumerator

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

5.16.4.49 OptixTraversableType

`enum OptixTraversableType`

Traversable Handles.

See also optixConvertPointerToTraversableHandle( )

Enumerator

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

5.16.4.50 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.

Enumerator

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)
- template<typename HitState >
  static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (const HitState &hs, float4 &m0, float4 &m1, float4 &m2)
- template<typename HitState >
  static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (const HitState &hs, float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ float3 optixTransformPoint (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &p)
- static __forceinline__ __device__ float3 optixTransformVector (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &v)
- static __forceinline__ __device__ float3 optixTransformNormal (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &n)
- OPTIX_MICROMAP_INLINE_FUNC float __uint_as_float (unsigned int x)

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

6.1.1 Function Documentation

6.1.1.1 optixAddFloat4()

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

6.1.1.2 optixGetInterpolatedTransformation() [1/2]

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

6.1.1.3 optixGetInterpolatedTransformation() [2/2]

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

6.1.1.4 optixGetInterpolatedTransformationFromHandle()

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

6.1.1.5 optixGetMatrixFromSrt()

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

6.1.1.6 optixGetObjectToWorldTransformMatrix()

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

6.1.1.7 optixGetWorldToObjectTransformMatrix()

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

6.1.1.8 optixInvertMatrix()

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

6.1.1.9 optixLdg()

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

6.1.1.10 optixLoadInterpolatedMatrixKey()

```
static __forceinline__ __device__ void optix_impl
::optixLoadInterpolatedMatrixKey (
          float4 & m0,
          float4 & m1,
          float4 & m2,
          const float4 * matrix,
```

```
          const float t1 )  [static]
```

6.1.1.11 optixLoadInterpolatedSrtKey()

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

6.1.1.12 optixLoadReadOnlyAlign16()

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

6.1.1.13 optixMulFloat4()

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

6.1.1.14 optixMultiplyRowMatrix()

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

6.1.1.15 optixResolveMotionKey()

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

6.1.1.16 optixTransformNormal()

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

```
          const float3 & n ) [static]
```

6.1.1.17 optixTransformPoint()

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

6.1.1.18 optixTransformVector()

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

6.2 optix_internal Namespace Reference

Classes

- struct TypePack

7 Class Documentation

7.1 OptixAabb Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.1.1 Detailed Description

AABB inputs.

7.1.2 Member Data Documentation

7.1.2.1 maxX

`float OptixAabb::maxX`

Upper extent in X direction.

7.1.2.2 maxY

`float OptixAabb::maxY`

Upper extent in Y direction.

7.1.2.3 maxZ

`float OptixAabb::maxZ`

Upper extent in Z direction.

7.1.2.4 minX

`float OptixAabb::minX`

Lower extent in X direction.

7.1.2.5 minY

`float OptixAabb::minY`

Lower extent in Y direction.

7.1.2.6 minZ

`float OptixAabb::minZ`

Lower extent in Z direction.

7.2 OptixAccelBufferSizes Struct Reference

`#include <optix_types.h>`

Public Attributes

- size_t outputSizeInBytes
- size_t tempSizeInBytes
- size_t tempUpdateSizeInBytes

7.2.1 Detailed Description

Struct for querying builder allocation requirements.

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

See also optixAccelComputeMemoryUsage( )

7.2.2 Member Data Documentation

7.2.2.1 outputSizeInBytes

`size_t OptixAccelBufferSizes::outputSizeInBytes`

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

7.2.2.2 tempSizeInBytes

`size_t OptixAccelBufferSizes::tempSizeInBytes`

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

7.2.2.3 tempUpdateSizeInBytes

`size_t OptixAccelBufferSizes::tempUpdateSizeInBytes`

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

7.3 OptixAccelBuildOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int buildFlags
- OptixBuildOperation operation
- OptixMotionOptions motionOptions

7.3.1 Detailed Description

Build options for acceleration structures.

See also optixAccelComputeMemoryUsage( ), optixAccelBuild( )

7.3.2 Member Data Documentation

7.3.2.1 buildFlags

`unsigned int OptixAccelBuildOptions::buildFlags`

Combinations of OptixBuildFlags.

7.3.2.2 motionOptions

`OptixMotionOptions OptixAccelBuildOptions::motionOptions`

Options for motion.

7.3.2.3 operation

`OptixBuildOperation OptixAccelBuildOptions::operation`

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

7.4 OptixAccelEmitDesc Struct Reference

`#include <optix_types.h>`

Public Attributes

- CUdeviceptr result
- OptixAccelPropertyType type

7.4.1 Detailed Description

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

See also optixAccelBuild( )

7.4.2 Member Data Documentation

7.4.2.1 result

`CUdeviceptr OptixAccelEmitDesc::result`

Output buffer for the properties.

7.4.2.2 type

`OptixAccelPropertyType OptixAccelEmitDesc::type`

Requested property.

7.5 OptixBuildInput Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixBuildInputType type
- union {
  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 OptixBuildInputInstanceArray Struct Reference

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

Public Attributes

- CUdeviceptr instances
- unsigned int numInstances
- unsigned int instanceStride

7.8.1 Detailed Description

Instance and instance pointer inputs.

See also OptixBuildInput::instanceArray

7.8.2 Member Data Documentation

7.8.2.1 instances

```
CUdeviceptr OptixBuildInputInstanceArray::instances
```

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

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

7.8.2.2 instanceStride

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

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

7.8.2.3 numInstances

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

Number of elements in OptixBuildInputInstanceArray::instances.

7.9 OptixBuildInputOpacityMicromap Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.9.1 Member Data Documentation

7.9.1.1 indexBuffer

`CUdeviceptr OptixBuildInputOpacityMicromap::indexBuffer`

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

7.9.1.2 indexingMode

`OptixOpacityMicromapArrayIndexingMode OptixBuildInputOpacityMicromap ::indexingMode`

Indexing mode of triangle to opacity micromap array mapping.

7.9.1.3 indexOffset

`unsigned int OptixBuildInputOpacityMicromap::indexOffset`

Constant offset to non-negative opacity micromap indices.

7.9.1.4 indexSizeInBytes

`unsigned int OptixBuildInputOpacityMicromap::indexSizeInBytes`

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

7.9.1.5 indexStrideInBytes

`unsigned int OptixBuildInputOpacityMicromap::indexStrideInBytes`

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

7.9.1.6 micromapUsageCounts

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

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

7.9.1.7 numMicromapUsageCounts

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

Number of OptixOpacityMicromapUsageCount.

7.9.1.8 opacityMicromapArray

```
CUdeviceptr OptixBuildInputOpacityMicromap::opacityMicromapArray
```

Device pointer to a opacity micromap array used by this build input array. This buffer is required when OptixBuildInputOpacityMicromap::indexingMode is OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR or OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED. Must be zero if OptixBuildInputOpacityMicromap::indexingMode is OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE.

7.10 OptixBuildInputSphereArray Struct Reference

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

Public Attributes

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

7.10.1 Detailed Description

Sphere inputs.

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

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

See also OptixBuildInput::sphereArray

7.10.2 Member Data Documentation

7.10.2.1 flags

```
const unsigned int* OptixBuildInputSphereArray::flags
```

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

7.10.2.2 numSbtRecords

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

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

7.10.2.3 numVertices

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

Number of vertices in each buffer in vertexBuffers.

7.10.2.4 primitiveIndexOffset

```
unsigned int OptixBuildInputSphereArray::primitiveIndexOffset
```

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

7.10.2.5 radiusBuffers

```
const CUdeviceptr* OptixBuildInputSphereArray::radiusBuffers
```

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

7.10.2.6 radiusStrideInBytes

```
unsigned int OptixBuildInputSphereArray::radiusStrideInBytes
```

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

7.10.2.7 sbtIndexOffsetBuffer

```
CUdeviceptr OptixBuildInputSphereArray::sbtIndexOffsetBuffer
```

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

7.10.2.8 sbtIndexOffsetSizeInBytes

```
unsigned int OptixBuildInputSphereArray::sbtIndexOffsetSizeInBytes
```

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

7.10.2.9 sbtIndexOffsetStrideInBytes

```
unsigned int OptixBuildInputSphereArray::sbtIndexOffsetStrideInBytes
```

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

7.10.2.10 singleRadius

```
int OptixBuildInputSphereArray::singleRadius
```

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

7.10.2.11 vertexBuffers

```
const CUdeviceptr* OptixBuildInputSphereArray::vertexBuffers
```

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

7.10.2.12 vertexStrideInBytes

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

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

7.11 OptixBuildInputTriangleArray Struct Reference

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

Public Attributes

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

7.11.1 Detailed Description

Triangle inputs.

See also OptixBuildInput::triangleArray

7.11.2 Member Data Documentation

7.11.2.1 flags

```
const unsigned int* OptixBuildInputTriangleArray::flags
```

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

7.11.2.2 indexBuffer

`CUdeviceptr OptixBuildInputTriangleArray::indexBuffer`

Optional pointer to array of 16 or 32-bit int triplets, one triplet per triangle. The minimum alignment must match the natural alignment of the type as specified in the indexFormat, i.e., for OPTIX_INDICES_FORMAT_UNSIGNED_INT3 4-byte and for OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 a 2-byte alignment.

7.11.2.3 indexFormat

`OptixIndicesFormat OptixBuildInputTriangleArray::indexFormat`

See also OptixIndicesFormat

7.11.2.4 indexStrideInBytes

`unsigned int OptixBuildInputTriangleArray::indexStrideInBytes`

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

7.11.2.5 numIndexTriplets

`unsigned int OptixBuildInputTriangleArray::numIndexTriplets`

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

7.11.2.6 numSbtRecords

`unsigned int OptixBuildInputTriangleArray::numSbtRecords`

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

7.11.2.7 numVertices

`unsigned int OptixBuildInputTriangleArray::numVertices`

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

7.11.2.8 opacityMicromap

`OptixBuildInputOpacityMicromap OptixBuildInputTriangleArray ::opacityMicromap`

Optional opacity micromap inputs.

7.11.2.9 preTransform

`CUdeviceptr OptixBuildInputTriangleArray::preTransform`

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

7.11.2.10 primitiveIndexOffset

`unsigned int OptixBuildInputTriangleArray::primitiveIndexOffset`

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

7.11.2.11 sbtIndexOffsetBuffer

`CUdeviceptr OptixBuildInputTriangleArray::sbtIndexOffsetBuffer`

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

7.11.2.12 sbtIndexOffsetSizeInBytes

`unsigned int OptixBuildInputTriangleArray::sbtIndexOffsetSizeInBytes`

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

7.11.2.13 sbtIndexOffsetStrideInBytes

`unsigned int OptixBuildInputTriangleArray::sbtIndexOffsetStrideInBytes`

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

7.11.2.14 transformFormat

`OptixTransformFormat OptixBuildInputTriangleArray::transformFormat`

See also OptixTransformFormat

7.11.2.15 vertexBuffers

`const CUdeviceptr* OptixBuildInputTriangleArray::vertexBuffers`

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

7.11.2.16 vertexFormat

`OptixVertexFormat OptixBuildInputTriangleArray::vertexFormat`

See also OptixVertexFormat

7.11.2.17 vertexStrideInBytes

`unsigned int OptixBuildInputTriangleArray::vertexStrideInBytes`

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

7.12 OptixBuiltinISOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.12.1 Detailed Description

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

See also optixBuiltinISModuleGet( )

7.12.2 Member Data Documentation

7.12.2.1 buildFlags

`unsigned int OptixBuiltinISOptions::buildFlags`

Build flags, see OptixBuildFlags.

7.12.2.2 builtinISModuleType

`OptixPrimitiveType OptixBuiltinISOptions::builtinISModuleType`

7.12.2.3 curveEndcapFlags

`unsigned int OptixBuiltinISOptions::curveEndcapFlags`

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

7.12.2.4 usesMotionBlur

`int OptixBuiltinISOptions::usesMotionBlur`

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

7.13 OptixClusterAccelBuildInput Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixClusterAccelBuildType type
- union {
  OptixClusterAccelBuildInputClusters clusters
  OptixClusterAccelBuildInputTriangles triangles
  OptixClusterAccelBuildInputGrids grids
  };

7.13.1 Member Data Documentation

7.13.1.1

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

7.13.1.2 clusters

```
OptixClusterAccelBuildInputClusters OptixClusterAccelBuildInput::clusters
```

7.13.1.3 grids

```
OptixClusterAccelBuildInputGrids OptixClusterAccelBuildInput::grids
```

7.13.1.4 triangles

```
OptixClusterAccelBuildInputTriangles OptixClusterAccelBuildInput::triangles
```

7.13.1.5 type

```
OptixClusterAccelBuildType OptixClusterAccelBuildInput::type
```

7.14 OptixClusterAccelBuildInputClusters Struct Reference

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

Public Attributes

- OptixClusterAccelBuildFlags flags
- unsigned int maxArgCount
- unsigned int maxTotalClusterCount
- unsigned int maxClusterCountPerArg

7.14.1 Member Data Documentation

7.14.1.1 flags

```
OptixClusterAccelBuildFlags OptixClusterAccelBuildInputClusters::flags
```

7.14.1.2 maxArgCount

```
unsigned int OptixClusterAccelBuildInputClusters::maxArgCount
```

7.14.1.3 maxClusterCountPerArg

```
unsigned int OptixClusterAccelBuildInputClusters::maxClusterCountPerArg
```

7.14.1.4 maxTotalClusterCount

```
unsigned int OptixClusterAccelBuildInputClusters::maxTotalClusterCount
```

7.15 OptixClusterAccelBuildInputClustersArgs Struct Reference

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

Public Attributes

- unsigned int clusterHandlesCount

- unsigned int clusterHandlesBufferStrideInBytes
- CUdeviceptr clusterHandlesBuffer

7.15.1 Detailed Description

Device data, args provided for OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS builds.

7.15.2 Member Data Documentation

7.15.2.1 clusterHandlesBuffer

```
CUdeviceptr OptixClusterAccelBuildInputClustersArgs::clusterHandlesBuffer
```

The clusterHandlesBuffer can come directly from CLAS builds output via OptixClusterAccelBuildModeDescImplicitDest::outputHandlesBuffer or OptixClusterAccelBuildModeDescExplicitDest::outputHandlesBuffer.

7.15.2.2 clusterHandlesBufferStrideInBytes

```
unsigned int OptixClusterAccelBuildInputClustersArgs
::clusterHandlesBufferStrideInBytes
```

7.15.2.3 clusterHandlesCount

```
unsigned int OptixClusterAccelBuildInputClustersArgs::clusterHandlesCount
```

7.16 OptixClusterAccelBuildInputGrids Struct Reference

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

Public Attributes

- OptixClusterAccelBuildFlags flags
- unsigned int maxArgCount
- OptixVertexFormat vertexFormat
- unsigned int maxSbtIndexValue
- unsigned int maxWidth
- unsigned int maxHeight

7.16.1 Member Data Documentation

7.16.1.1 flags

```
OptixClusterAccelBuildFlags OptixClusterAccelBuildInputGrids::flags
```

7.16.1.2 maxArgCount

```
unsigned int OptixClusterAccelBuildInputGrids::maxArgCount
```

7.16.1.3 maxHeight

```
unsigned int OptixClusterAccelBuildInputGrids::maxHeight
```

7.16.1.4 maxSbtIndexValue

```
unsigned int OptixClusterAccelBuildInputGrids::maxSbtIndexValue
```

7.16.1.5 maxWidth

```
unsigned int OptixClusterAccelBuildInputGrids::maxWidth
```

7.16.1.6 vertexFormat

```
OptixVertexFormat OptixClusterAccelBuildInputGrids::vertexFormat
```

7.17 OptixClusterAccelBuildInputGridsArgs Struct Reference

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

Public Attributes

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

7.17.1 Detailed Description

Device data, args provided for OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS builds.

7.17.2 Member Data Documentation

7.17.2.1 baseClusterId

```
unsigned int OptixClusterAccelBuildInputGridsArgs::baseClusterId
```

7.17.2.2 basePrimitiveInfo

```
OptixClusterAccelPrimitiveInfo OptixClusterAccelBuildInputGridsArgs
::basePrimitiveInfo
```

7.17.2.3 clusterFlags

```
unsigned int OptixClusterAccelBuildInputGridsArgs::clusterFlags
```

7.17.2.4 dimensions

```
unsigned char OptixClusterAccelBuildInputGridsArgs::dimensions[2]
```

7.17.2.5 positionTruncateBitCount

```
unsigned int OptixClusterAccelBuildInputGridsArgs::positionTruncateBitCount
```

7.17.2.6 reserved

```
unsigned int OptixClusterAccelBuildInputGridsArgs::reserved
```

7.17.2.7 reserved2

```
unsigned short OptixClusterAccelBuildInputGridsArgs::reserved2
```

7.18 OptixClusterAccelBuildInputTemplatesArgs Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int clusterIdOffset
- unsigned int sbtIndexOffset
- CUdeviceptr clusterTemplate
- CUdeviceptr vertexBuffer
- unsigned int vertexStrideInBytes
- unsigned int reserved

7.18.1 Detailed Description

Device data, args provided for OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES builds.

7.18.2 Member Data Documentation

7.18.2.1 clusterIdOffset

```
unsigned int OptixClusterAccelBuildInputTemplatesArgs::clusterIdOffset
```

7.18.2.2 clusterTemplate

```
CUdeviceptr OptixClusterAccelBuildInputTemplatesArgs::clusterTemplate
```

7.18.2.3 reserved

```
unsigned int OptixClusterAccelBuildInputTemplatesArgs::reserved
```

7.18.2.4 sbtIndexOffset

```
unsigned int OptixClusterAccelBuildInputTemplatesArgs::sbtIndexOffset
```

7.18.2.5 vertexBuffer

```
CUdeviceptr OptixClusterAccelBuildInputTemplatesArgs::vertexBuffer
```

7.18.2.6 vertexStrideInBytes

```
unsigned int OptixClusterAccelBuildInputTemplatesArgs::vertexStrideInBytes
```

7.19 OptixClusterAccelBuildInputTriangles Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixClusterAccelBuildFlags flags
- unsigned int maxArgCount
- OptixVertexFormat vertexFormat
- unsigned int maxSbtIndexValue
- unsigned int maxUniqueSbtIndexCountPerArg
- unsigned int maxTriangleCountPerArg

- unsigned int maxVertexCountPerArg
- unsigned int maxTotalTriangleCount
- unsigned int maxTotalVertexCount
- unsigned int minPositionTruncateBitCount

7.19.1 Member Data Documentation

7.19.1.1 flags

```
OptixClusterAccelBuildFlags OptixClusterAccelBuildInputTriangles::flags
```

7.19.1.2 maxArgCount

```
unsigned int OptixClusterAccelBuildInputTriangles::maxArgCount
```

max number of OptixClusterAccelBuildInputTrianglesArgs provided at build time for OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES and OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES max number of OptixClusterAccelBuildInputTemplatesArgs provided at build time for OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES

7.19.1.3 maxSbtIndexValue

```
unsigned int OptixClusterAccelBuildInputTriangles::maxSbtIndexValue
```

7.19.1.4 maxTotalTriangleCount

```
unsigned int OptixClusterAccelBuildInputTriangles::maxTotalTriangleCount
```

7.19.1.5 maxTotalVertexCount

```
unsigned int OptixClusterAccelBuildInputTriangles::maxTotalVertexCount
```

7.19.1.6 maxTriangleCountPerArg

```
unsigned int OptixClusterAccelBuildInputTriangles::maxTriangleCountPerArg
```

7.19.1.7 maxUniqueSbtIndexCountPerArg

```
unsigned int OptixClusterAccelBuildInputTriangles
::maxUniqueSbtIndexCountPerArg
```

7.19.1.8 maxVertexCountPerArg

```
unsigned int OptixClusterAccelBuildInputTriangles::maxVertexCountPerArg
```

7.19.1.9 minPositionTruncateBitCount

```
unsigned int OptixClusterAccelBuildInputTriangles
::minPositionTruncateBitCount
```

7.19.1.10 vertexFormat

```
OptixVertexFormat OptixClusterAccelBuildInputTriangles::vertexFormat
```

7.20 OptixClusterAccelBuildInputTrianglesArgs Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.20.1 Detailed Description

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

7.20.2 Member Data Documentation

7.20.2.1 basePrimitiveInfo

```
OptixClusterAccelPrimitiveInfo OptixClusterAccelBuildInputTrianglesArgs
::basePrimitiveInfo
```

7.20.2.2 clusterFlags

```
unsigned int OptixClusterAccelBuildInputTrianglesArgs::clusterFlags
```

7.20.2.3 clusterId

```
unsigned int OptixClusterAccelBuildInputTrianglesArgs::clusterId
```

7.20.2.4 indexBuffer

```
CUdeviceptr OptixClusterAccelBuildInputTrianglesArgs::indexBuffer
```

7.20.2.5 indexBufferStrideInBytes

```
unsigned short OptixClusterAccelBuildInputTrianglesArgs
::indexBufferStrideInBytes
```

7.20.2.6 indexFormat

```
unsigned int OptixClusterAccelBuildInputTrianglesArgs::indexFormat
```

7.20.2.7 instantiationBoundingBoxLimit

```
CUdeviceptr OptixClusterAccelBuildInputTrianglesArgs
::instantiationBoundingBoxLimit
```

Optional with OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES, 32-byte-aligned pointer to OptixAabb, one per cluster, limiting the extent of each cluster Vertices provided for template instantiation must not be outside the bounding box. Providing a bounding box may improve compression (reduced CLAS size) as well as trace performance. Ignored for OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES.

7.20.2.8 opacityMicromapArray

```
CUdeviceptr OptixClusterAccelBuildInputTrianglesArgs::opacityMicromapArray
```

7.20.2.9 opacityMicromapIndexBuffer

```
CUdeviceptr OptixClusterAccelBuildInputTrianglesArgs
::opacityMicromapIndexBuffer
```

7.20.2.10 opacityMicromapIndexBufferStrideInBytes

```
unsigned short OptixClusterAccelBuildInputTrianglesArgs
::opacityMicromapIndexBufferStrideInBytes
```

7.20.2.11 opacityMicromapIndexFormat

```
unsigned int OptixClusterAccelBuildInputTrianglesArgs
::opacityMicromapIndexFormat
```

7.20.2.12 positionTruncateBitCount

```
unsigned int OptixClusterAccelBuildInputTrianglesArgs
::positionTruncateBitCount
```

Number of LSB in mantissa that are dropped (0 means don't drop any) for float32 positions. Other formats are first converted to float32 before dropping bits. Builder will drop bits when building CLAS / instantiating cluster templates (no need to truncate the input before build).

7.20.2.13 primitiveInfoBuffer

```
CUdeviceptr OptixClusterAccelBuildInputTrianglesArgs::primitiveInfoBuffer
```

7.20.2.14 primitiveInfoBufferStrideInBytes

```
unsigned short OptixClusterAccelBuildInputTrianglesArgs
::primitiveInfoBufferStrideInBytes
```

7.20.2.15 triangleCount

```
unsigned int OptixClusterAccelBuildInputTrianglesArgs::triangleCount
```

7.20.2.16 vertexBuffer

```
CUdeviceptr OptixClusterAccelBuildInputTrianglesArgs::vertexBuffer
```

vertexBuffer is mandatory when using OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES. Optional with OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES and when specified provide example "hint" vertices for templates; actual vertices are specified at template instantiation. It is typically useful to provide vertices for template creation in scenarios such as animation, where the relative locality of vertices is expected to be similar between the template creation and instantiation

7.20.2.17 vertexBufferStrideInBytes

```
unsigned short OptixClusterAccelBuildInputTrianglesArgs
::vertexBufferStrideInBytes
```

7.20.2.18 vertexCount

```
unsigned int OptixClusterAccelBuildInputTrianglesArgs::vertexCount
```

7.21 OptixClusterAccelBuildModeDesc Struct Reference

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

Public Attributes

- OptixClusterAccelBuildMode mode
- union {
  OptixClusterAccelBuildModeDescImplicitDest implicitDest
  OptixClusterAccelBuildModeDescExplicitDest explicitDest
  OptixClusterAccelBuildModeDescGetSize getSize
  };

7.21.1 Member Data Documentation

7.21.1.1

```
union { ... } OptixClusterAccelBuildModeDesc::@7
```

7.21.1.2 explicitDest

```
OptixClusterAccelBuildModeDescExplicitDest OptixClusterAccelBuildModeDesc
::explicitDest
```

7.21.1.3 getSize

```
OptixClusterAccelBuildModeDescGetSize OptixClusterAccelBuildModeDesc
::getSize
```

7.21.1.4 implicitDest

```
OptixClusterAccelBuildModeDescImplicitDest OptixClusterAccelBuildModeDesc
::implicitDest
```

7.21.1.5 mode

```
OptixClusterAccelBuildMode OptixClusterAccelBuildModeDesc::mode
```

7.22 OptixClusterAccelBuildModeDescExplicitDest Struct Reference

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

Public Attributes

- CUdeviceptr tempBuffer
- size_t tempBufferSizeInBytes
- CUdeviceptr destAddressesBuffer
- unsigned int destAddressesStrideInBytes
- CUdeviceptr outputHandlesBuffer
- unsigned int outputHandlesStrideInBytes
- CUdeviceptr outputSizesBuffer
- unsigned int outputSizesStrideInBytes

7.22.1 Member Data Documentation

7.22.1.1 destAddressesBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescExplicitDest::destAddressesBuffer
```

7.22.1.2 destAddressesStrideInBytes

```
unsigned int OptixClusterAccelBuildModeDescExplicitDest
::destAddressesStrideInBytes
```

7.22.1.3 outputHandlesBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescExplicitDest::outputHandlesBuffer
```

7.22.1.4 outputHandlesStrideInBytes

```
unsigned int OptixClusterAccelBuildModeDescExplicitDest
::outputHandlesStrideInBytes
```

7.22.1.5 outputSizesBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescExplicitDest::outputSizesBuffer
```

7.22.1.6 outputSizesStrideInBytes

```
unsigned int OptixClusterAccelBuildModeDescExplicitDest
::outputSizesStrideInBytes
```

7.22.1.7 tempBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescExplicitDest::tempBuffer
```

7.22.1.8 tempBufferSizeInBytes

```
size_t OptixClusterAccelBuildModeDescExplicitDest::tempBufferSizeInBytes
```

7.23 OptixClusterAccelBuildModeDescGetSize Struct Reference

`#include <optix_types.h>`

Public Attributes

- CUdeviceptr outputSizesBuffer
- unsigned int outputSizesStrideInBytes
- CUdeviceptr tempBuffer
- size_t tempBufferSizeInBytes

7.23.1 Member Data Documentation

7.23.1.1 outputSizesBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescGetSize::outputSizesBuffer
```

7.23.1.2 outputSizesStrideInBytes

```
unsigned int OptixClusterAccelBuildModeDescGetSize::outputSizesStrideInBytes
```

7.23.1.3 tempBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescGetSize::tempBuffer
```

7.23.1.4 tempBufferSizeInBytes

```
size_t OptixClusterAccelBuildModeDescGetSize::tempBufferSizeInBytes
```

7.24 OptixClusterAccelBuildModeDescImplicitDest Struct Reference

`#include <optix_types.h>`

Public Attributes

- CUdeviceptr outputBuffer
- size_t outputBufferSizeInBytes
- CUdeviceptr tempBuffer
- size_t tempBufferSizeInBytes
- CUdeviceptr outputHandlesBuffer
- unsigned int outputHandlesStrideInBytes
- CUdeviceptr outputSizesBuffer
- unsigned int outputSizesStrideInBytes

7.24.1 Member Data Documentation

7.24.1.1 outputBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescImplicitDest::outputBuffer
```

alignment of outputBuffer must match result type. Clusters: 128 bytes Templates: 32 bytes GASes: 128 bytes, see OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT

7.24.1.2 outputBufferSizeInBytes

```
size_t OptixClusterAccelBuildModeDescImplicitDest::outputBufferSizeInBytes
```

7.24.1.3 outputHandlesBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescImplicitDest::outputHandlesBuffer
```

7.24.1.4 outputHandlesStrideInBytes

```
unsigned int OptixClusterAccelBuildModeDescImplicitDest
::outputHandlesStrideInBytes
```

7.24.1.5 outputSizesBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescImplicitDest::outputSizesBuffer
```

7.24.1.6 outputSizesStrideInBytes

```
unsigned int OptixClusterAccelBuildModeDescImplicitDest
::outputSizesStrideInBytes
```

7.24.1.7 tempBuffer

```
CUdeviceptr OptixClusterAccelBuildModeDescImplicitDest::tempBuffer
```

7.24.1.8 tempBufferSizeInBytes

```
size_t OptixClusterAccelBuildModeDescImplicitDest::tempBufferSizeInBytes
```

7.25 OptixClusterAccelPrimitiveInfo Struct Reference

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

Public Attributes

- unsigned int sbtIndex: 24
- unsigned int reserved: 5
- unsigned int primitiveFlags: 3

7.25.1 Member Data Documentation

7.25.1.1 primitiveFlags

```
unsigned int OptixClusterAccelPrimitiveInfo::primitiveFlags
```

7.25.1.2 reserved

```
unsigned int OptixClusterAccelPrimitiveInfo::reserved
```

7.25.1.3 sbtIndex

```
unsigned int OptixClusterAccelPrimitiveInfo::sbtIndex
```

7.26 OptixCoopVec< T, N > Class Template Reference

```
#include <optix_device.h>
```

Public Types

- using value_type = T

Public Member Functions

- __forceinline__ __device__ OptixCoopVec ()
- __forceinline__ __device__ OptixCoopVec (const value_type &val)
- __forceinline__ __device__ const value_type & operator[ ] (unsigned int index) const
- __forceinline__ __device__ value_type & operator[ ] (unsigned int index)
- __forceinline__ __device__ const value_type * data () const
- __forceinline__ __device__ value_type * data ()

Static Public Attributes

- static const unsigned int size = N

Protected Attributes

- value_type m_data [size]

7.26.1 Detailed Description

template<typename T, unsigned int N>

class OptixCoopVec< T, N >

The API does not require the use of this class specifically, but it must define a certain interface as spelled out by the public members of the class. Note that not all types of T are supported. Only 8 and 32 bit signed and unsigned integral types along with 16 and 32 bit floating point values.

7.26.2 Member Typedef Documentation

7.26.2.1 value_type

```
template<typename T , unsigned int N>
using OptixCoopVec< T, N >::value_type = T
```

7.26.3 Constructor & Destructor Documentation

7.26.3.1 OptixCoopVec() [1/2]

```
template<typename T , unsigned int N>
__forceinline__ __device__ OptixCoopVec< T, N >::OptixCoopVec ( )  [inline]
```

7.26.3.2 OptixCoopVec() [2/2]

```
template<typename T , unsigned int N>
__forceinline__ __device__ OptixCoopVec< T, N >::OptixCoopVec (
          const value_type & val )  [inline]
```

7.26.4 Member Function Documentation

7.26.4.1 data() [1/2]

```
template<typename T , unsigned int N>
__forceinline__ __device__ value_type * OptixCoopVec< T, N >::data ( )
[inline]
```

7.26.4.2 data() [2/2]

```
template<typename T , unsigned int N>
__forceinline__ __device__ const value_type * OptixCoopVec< T, N >::data (
) const [inline]
```

7.26.4.3 operator[]() [1/2]

```
template<typename T , unsigned int N>
__forceinline__ __device__ value_type & OptixCoopVec< T, N >::operator[ ] (
          unsigned int index ) [inline]
```

7.26.4.4 operator[]() [2/2]

```
template<typename T , unsigned int N>
__forceinline__ __device__ const value_type & OptixCoopVec< T, N >
::operator[ ] (
          unsigned int index ) const [inline]
```

7.26.5 Member Data Documentation

7.26.5.1 m_data

```
template<typename T , unsigned int N>
value_type OptixCoopVec< T, N >::m_data[size] [protected]
```

7.26.5.2 size

```
template<typename T , unsigned int N>
const unsigned int OptixCoopVec< T, N >::size = N [static]
```

7.27 OptixCoopVecMatrixDescription Struct Reference

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

Public Attributes

- unsigned int N
- unsigned int K
- unsigned int offsetInBytes
- OptixCoopVecElemType elementType
- OptixCoopVecMatrixLayout layout
- unsigned int rowColumnStrideInBytes
- unsigned int sizeInBytes

7.27.1 Detailed Description

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

The rowColumnStrideInBytes is ignored if the layout is either OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL or OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL

7.27.2 Member Data Documentation

7.27.2.1 elementType

```
OptixCoopVecElemType OptixCoopVecMatrixDescription::elementType
```

7.27.2.2 K

```
unsigned int OptixCoopVecMatrixDescription::K
```

7.27.2.3 layout

```
OptixCoopVecMatrixLayout OptixCoopVecMatrixDescription::layout
```

7.27.2.4 N

```
unsigned int OptixCoopVecMatrixDescription::N
```

7.27.2.5 offsetInBytes

```
unsigned int OptixCoopVecMatrixDescription::offsetInBytes
```

7.27.2.6 rowColumnStrideInBytes

```
unsigned int OptixCoopVecMatrixDescription::rowColumnStrideInBytes
```

7.27.2.7 sizeInBytes

```
unsigned int OptixCoopVecMatrixDescription::sizeInBytes
```

7.28 OptixDenoiserGuideLayer Struct Reference

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

Public Attributes

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

7.28.1 Detailed Description

Guide layer for the denoiser.

See also optixDenoiserInvoke( )

7.28.2 Member Data Documentation

7.28.2.1 albedo

```
OptixImage2D OptixDenoiserGuideLayer::albedo
```

7.28.2.2 flow

```
OptixImage2D OptixDenoiserGuideLayer::flow
```

7.28.2.3 flowTrustworthiness

```
OptixImage2D OptixDenoiserGuideLayer::flowTrustworthiness
```

7.28.2.4 normal

```
OptixImage2D OptixDenoiserGuideLayer::normal
```

7.28.2.5 outputInternalGuideLayer

```
OptixImage2D OptixDenoiserGuideLayer::outputInternalGuideLayer
```

7.28.2.6 previousOutputInternalGuideLayer

```
OptixImage2D OptixDenoiserGuideLayer::previousOutputInternalGuideLayer
```

7.29 OptixDenoiserLayer Struct Reference

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

Public Attributes

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

7.29.1 Detailed Description

Input/Output layers for the denoiser.

See also optixDenoiserInvoke( )

7.29.2 Member Data Documentation

7.29.2.1 input

```
OptixImage2D OptixDenoiserLayer::input
```

7.29.2.2 output

```
OptixImage2D OptixDenoiserLayer::output
```

7.29.2.3 previousOutput

```
OptixImage2D OptixDenoiserLayer::previousOutput
```

7.29.2.4 type

```
OptixDenoiserAOVType OptixDenoiserLayer::type
```

7.30 OptixDenoiserOptions Struct Reference

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

Public Attributes

- unsigned int guideAlbedo

- unsigned int guideNormal
- OptixDenoiserAlphaMode denoiseAlpha

7.30.1 Detailed Description

Options used by the denoiser.

See also optixDenoiserCreate( )

7.30.2 Member Data Documentation

7.30.2.1 denoiseAlpha

`OptixDenoiserAlphaMode OptixDenoiserOptions::denoiseAlpha`

alpha denoise mode

7.30.2.2 guideAlbedo

`unsigned int OptixDenoiserOptions::guideAlbedo`

7.30.2.3 guideNormal

`unsigned int OptixDenoiserOptions::guideNormal`

7.31 OptixDenoiserParams Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.31.1 Detailed Description

Various parameters used by the denoiser.

See also optixDenoiserInvoke( )

optixDenoiserComputeIntensity( )

optixDenoiserComputeAverageColor( )

7.31.2 Member Data Documentation

7.31.2.1 blendFactor

`float OptixDenoiserParams::blendFactor`

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

7.31.2.2 hdrAverageColor

`CUdeviceptr OptixDenoiserParams::hdrAverageColor`

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

7.31.2.3 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.

7.31.2.4 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.

7.32 OptixDenoiserSizes Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.32.1 Detailed Description

Various sizes related to the denoiser.

See also optixDenoiserComputeMemoryResources( )

7.32.2 Member Data Documentation

7.32.2.1 computeAverageColorSizeInBytes

`size_t OptixDenoiserSizes::computeAverageColorSizeInBytes`

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

7.32.2.2 computeIntensitySizeInBytes

`size_t OptixDenoiserSizes::computeIntensitySizeInBytes`

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

7.32.2.3 internalGuideLayerPixelSizeInBytes

`size_t OptixDenoiserSizes::internalGuideLayerPixelSizeInBytes`

Number of bytes for each pixel in internal guide layers.

7.32.2.4 overlapWindowSizeInPixels

`unsigned int OptixDenoiserSizes::overlapWindowSizeInPixels`

Overlap on all four tile sides.

7.32.2.5 stateSizeInBytes

`size_t OptixDenoiserSizes::stateSizeInBytes`

Size of state memory passed to optixDenoiserSetup, optixDenoiserInvoke.

7.32.2.6 withoutOverlapScratchSizeInBytes

`size_t OptixDenoiserSizes::withoutOverlapScratchSizeInBytes`

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

7.32.2.7 withOverlapScratchSizeInBytes

`size_t OptixDenoiserSizes::withOverlapScratchSizeInBytes`

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

7.33 OptixDeviceContextOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.33.1 Detailed Description

Parameters used for optixDeviceContextCreate( )

See also optixDeviceContextCreate( )

7.33.2 Member Data Documentation

7.33.2.1 logCallbackData

`void* OptixDeviceContextOptions::logCallbackData`

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

7.33.2.2 logCallbackFunction

`OptixLogCallback OptixDeviceContextOptions::logCallbackFunction`

Function pointer used when OptiX wishes to generate messages.

7.33.2.3 logCallbackLevel

`int OptixDeviceContextOptions::logCallbackLevel`

Maximum callback level to generate message for (see OptixLogCallback)

7.33.2.4 validationMode

`OptixDeviceContextValidationMode OptixDeviceContextOptions::validationMode`

Validation mode of context.

7.34 OptixFunctionTable Struct Reference

`#include <optix_function_table.h>`

Public Attributes

Error handling

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

Device context

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

Modules

- OptixResult(* optixModuleCreate)(OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module)
- OptixResult(* optixModuleCreateWithTasks)(OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module, OptixTask *firstTask)
- OptixResult(* optixModuleGetCompilationState)(OptixModule module, OptixModuleCompileState *state)
- OptixResult(* 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(∗ stub1)(void)
- OptixResult(∗ stub2)(void)
- OptixResult(∗ optixClusterAccelComputeMemoryUsage)(OptixDeviceContext context, OptixClusterAccelBuildMode buildMode, const OptixClusterAccelBuildInput ∗buildInput, OptixAccelBufferSizes ∗bufferSizes)
- OptixResult(∗ optixClusterAccelBuild)(OptixDeviceContext context, CUstream stream, const OptixClusterAccelBuildModeDesc ∗buildModeDesc, const OptixClusterAccelBuildInput ∗buildInput, CUdeviceptr argsArray, CUdeviceptr argsCount, unsigned int argsStrideInBytes)

Launch

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

Cooperative Vector

- OptixResult(∗ optixCoopVecMatrixConvert)(OptixDeviceContext context, CUstream stream, unsigned int numNetworks, const OptixNetworkDescription ∗inputNetworkDescription, CUdeviceptr inputNetworks, size_t inputNetworkStrideInBytes, const OptixNetworkDescription ∗outputNetworkDescription, CUdeviceptr outputNetworks, size_t outputNetworkStrideInBytes)
- OptixResult(∗ optixCoopVecMatrixComputeSize)(OptixDeviceContext context, unsigned int N, unsigned int K, OptixCoopVecElemType elementType, OptixCoopVecMatrixLayout layout, size_t rowColumnStrideInBytes, size_t ∗sizeInBytes)

Denoiser

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

7.34.1 Detailed Description

The function table containing all API functions.

See optixInit( ) and optixInitWithHandle( ).

7.34.2 Member Data Documentation

7.34.2.1 optixAccelBuild

```
OptixResult(* OptixFunctionTable::optixAccelBuild) (OptixDeviceContext
context, CUstream stream, const OptixAccelBuildOptions *accelOptions, const
OptixBuildInput *buildInputs, unsigned int numBuildInputs, CUdeviceptr
tempBuffer, size_t tempBufferSizeInBytes, CUdeviceptr outputBuffer, size_t
outputBufferSizeInBytes, OptixTraversableHandle *outputHandle, const
OptixAccelEmitDesc *emittedProperties, unsigned int numEmittedProperties)
```

See optixAccelBuild( ).

7.34.2.2 optixAccelCompact

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

See optixAccelCompact( ).

7.34.2.3 optixAccelComputeMemoryUsage

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

See optixAccelComputeMemoryUsage( ).

7.34.2.4 optixAccelEmitProperty

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

See optixAccelComputeMemoryUsage( ).

7.34.2.5 optixAccelGetRelocationInfo

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

See optixAccelGetRelocationInfo( ).

7.34.2.6 optixAccelRelocate

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

```
*targetHandle)
```

See optixAccelRelocate( ).

7.34.2.7 optixBuiltinISModuleGet

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

See optixBuiltinISModuleGet( ).

7.34.2.8 optixCheckRelocationCompatibility

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

See optixCheckRelocationCompatibility( ).

7.34.2.9 optixClusterAccelBuild

```
OptixResult(* OptixFunctionTable::optixClusterAccelBuild)
(OptixDeviceContext context, CUstream stream, const
OptixClusterAccelBuildModeDesc *buildModeDesc, const
OptixClusterAccelBuildInput *buildInput, CUdeviceptr argsArray, CUdeviceptr
argsCount, unsigned int argsStrideInBytes)
```

See optixClusterAccelBuild( ).

7.34.2.10 optixClusterAccelComputeMemoryUsage

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

See optixClusterAccelComputeMemoryUsage( ).

7.34.2.11 optixConvertPointerToTraversableHandle

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

See optixConvertPointerToTraversableHandle( ).

7.34.2.12 optixCoopVecMatrixComputeSize

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

See optixCoopVecMatrixComputeSize( ).

7.34.2.13 optixCoopVecMatrixConvert

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

See optixCoopVecMatrixConvert( ).

7.34.2.14 optixDenoiserComputeAverageColor

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

See optixDenoiserComputeAverageColor( ).

7.34.2.15 optixDenoiserComputeIntensity

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

See optixDenoiserComputeIntensity( ).

7.34.2.16 optixDenoiserComputeMemoryResources

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

See optixDenoiserComputeMemoryResources( ).

7.34.2.17 optixDenoiserCreate

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

See optixDenoiserCreate( ).

7.34.2.18 optixDenoiserCreateWithUserModel

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

See optixDenoiserCreateWithUserModel( ).

7.34.2.19 optixDenoiserDestroy

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

See optixDenoiserDestroy( ).

7.34.2.20 optixDenoiserInvoke

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

See optixDenoiserInvoke( ).

7.34.2.21 optixDenoiserSetup

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

See optixDenoiserSetup( ).

7.34.2.22 optixDeviceContextCreate

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

See optixDeviceContextCreate( ).

7.34.2.23 optixDeviceContextDestroy

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

See optixDeviceContextDestroy( ).

7.34.2.24 optixDeviceContextGetCacheDatabaseSizes

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

See optixDeviceContextGetCacheDatabaseSizes( ).

7.34.2.25 optixDeviceContextGetCacheEnabled

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

See optixDeviceContextGetCacheEnabled( ).

7.34.2.26 optixDeviceContextGetCacheLocation

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

See optixDeviceContextGetCacheLocation( ).

7.34.2.27 optixDeviceContextGetProperty

```
OptixResult(* OptixFunctionTable::optixDeviceContextGetProperty)
```

```
(OptixDeviceContext context, OptixDeviceProperty property, void *value, size
_t sizeInBytes)
```

See optixDeviceContextGetProperty( ).

7.34.2.28 optixDeviceContextSetCacheDatabaseSizes

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

See optixDeviceContextSetCacheDatabaseSizes( ).

7.34.2.29 optixDeviceContextSetCacheEnabled

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

See optixDeviceContextSetCacheEnabled( ).

7.34.2.30 optixDeviceContextSetCacheLocation

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

See optixDeviceContextSetCacheLocation( ).

7.34.2.31 optixDeviceContextSetLogCallback

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

See optixDeviceContextSetLogCallback( ).

7.34.2.32 optixGetErrorName

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

See optixGetErrorName( ).

7.34.2.33 optixGetErrorString

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

See optixGetErrorString( ).

7.34.2.34 optixLaunch

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

See optixConvertPointerToTraversableHandle( ).

7.34.2.35 optixModuleCreate

```
OptixResult(* OptixFunctionTable::optixModuleCreate) (OptixDeviceContext
context, const OptixModuleCompileOptions *moduleCompileOptions, const
OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size
```

```
_t inputSize, char *logString, size_t *logStringSize, OptixModule *module)
```

See optixModuleCreate( ).

7.34.2.36 optixModuleCreateWithTasks

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

See optixModuleCreateWithTasks( ).

7.34.2.37 optixModuleDestroy

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

See optixModuleDestroy( ).

7.34.2.38 optixModuleGetCompilationState

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

See optixModuleGetCompilationState( ).

7.34.2.39 optixOpacityMicromapArrayBuild

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

See optixOpacityMicromapArrayBuild( ).

7.34.2.40 optixOpacityMicromapArrayComputeMemoryUsage

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

See optixOpacityMicromapArrayComputeMemoryUsage( ).

7.34.2.41 optixOpacityMicromapArrayGetRelocationInfo

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

See optixOpacityMicromapArrayGetRelocationInfo( ).

7.34.2.42 optixOpacityMicromapArrayRelocate

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

```
targetOpacityMicromapArraySizeInBytes)
```

See optixOpacityMicromapArrayRelocate( ).

7.34.2.43 optixPipelineCreate

```
OptixResult(* OptixFunctionTable::optixPipelineCreate) (OptixDeviceContext
context, const OptixPipelineCompileOptions *pipelineCompileOptions, const
OptixPipelineLinkOptions *pipelineLinkOptions, const OptixProgramGroup
*programGroups, unsigned int numProgramGroups, char *logString, size_t
*logStringSize, OptixPipeline *pipeline)
```

See optixPipelineCreate( ).

7.34.2.44 optixPipelineDestroy

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

See optixPipelineDestroy( ).

7.34.2.45 optixPipelineSetStackSize

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

See optixPipelineSetStackSize( ).

7.34.2.46 optixProgramGroupCreate

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

See optixProgramGroupCreate( ).

7.34.2.47 optixProgramGroupDestroy

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

See optixProgramGroupDestroy( ).

7.34.2.48 optixProgramGroupGetStackSize

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

See optixProgramGroupGetStackSize( ).

7.34.2.49 optixSbtRecordPackHeader

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

See optixConvertPointerToTraversableHandle( ).

7.34.2.50 optixTaskExecute

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

See optixTaskExecute( ).

7.34.2.51 stub1

```
OptixResult(* OptixFunctionTable::stub1) (void)
```

See optixAccelComputeMemoryUsage( ).

7.34.2.52 stub2

```
OptixResult(* OptixFunctionTable::stub2) (void)
```

See optixAccelComputeMemoryUsage( ).

7.35 OptixImage2D Struct Reference

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

Public Attributes

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

7.35.1 Detailed Description

Image descriptor used by the denoiser.

See also optixDenoiserInvoke( ), optixDenoiserComputeIntensity( )

7.35.2 Member Data Documentation

7.35.2.1 data

```
CUdeviceptr OptixImage2D::data
```

Pointer to the actual pixel data.

7.35.2.2 format

```
OptixPixelFormat OptixImage2D::format
```

Pixel format.

7.35.2.3 height

```
unsigned int OptixImage2D::height
```

Height of the image (in pixels)

7.35.2.4 pixelStrideInBytes

`unsigned int OptixImage2D::pixelStrideInBytes`

Stride between subsequent pixels of the image (in bytes). If set to 0, dense packing (no gaps) is assumed. For pixel format OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER it must be set to OptixDenoiserSizes::internalGuideLayerPixelSizeInBytes.

7.35.2.5 rowStrideInBytes

`unsigned int OptixImage2D::rowStrideInBytes`

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

7.35.2.6 width

`unsigned int OptixImage2D::width`

Width of the image (in pixels)

7.36 OptixIncomingHitObject Struct Reference

`#include <optix_device.h>`

Public Member Functions

- __forceinline__ __device__ float getRayTime () const
- __forceinline__ __device__ unsigned int getTransformListSize () const
- __forceinline__ __device__ OptixTraversableHandle getTransformListHandle (unsigned int index) const

7.36.1 Member Function Documentation

7.36.1.1 getRayTime()

`__forceinline__ __device__ float OptixIncomingHitObject::getRayTime ( ) const` *[inline]*

7.36.1.2 getTransformListHandle()

```
__forceinline__ __device__ OptixTraversableHandle OptixIncomingHitObject
::getTransformListHandle (
          unsigned int index ) const [inline]
```

7.36.1.3 getTransformListSize()

```
__forceinline__ __device__ unsigned int OptixIncomingHitObject
::getTransformListSize ( ) const [inline]
```

7.37 OptixInstance Struct Reference

`#include <optix_types.h>`

Public Attributes

- float transform [12]
- unsigned int instanceId
- unsigned int sbtOffset

- unsigned int visibilityMask
- unsigned int flags
- OptixTraversableHandle traversableHandle
- unsigned int pad [2]

7.37.1 Detailed Description

Instances.

See also OptixBuildInputInstanceArray::instances This struct is interpreted on the device by rtcore, and should mirror the RtcFatInstance.

7.37.2 Member Data Documentation

7.37.2.1 flags

`unsigned int OptixInstance::flags`

Any combination of OptixInstanceFlags is allowed.

7.37.2.2 instanceId

`unsigned int OptixInstance::instanceId`

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

7.37.2.3 pad

`unsigned int OptixInstance::pad[2]`

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

7.37.2.4 sbtOffset

`unsigned int OptixInstance::sbtOffset`

SBT record offset. In a traversable graph with multiple levels of instance acceleration structure (IAS) objects, offsets are summed together. The maximal SBT offset can be queried using OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET.

7.37.2.5 transform

`float OptixInstance::transform[12]`

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

7.37.2.6 traversableHandle

`OptixTraversableHandle OptixInstance::traversableHandle`

Set with an OptixTraversableHandle.

7.37.2.7 visibilityMask

`unsigned int OptixInstance::visibilityMask`

Visibility mask. If rayMask & instanceMask == 0 the instance is culled. The number of available bits can be queried using OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK.

7.38 OptixMatrixMotionTransform Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.38.1 Detailed Description

Represents a matrix motion transformation.

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

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

```
float matrixData[N][12];
... // setup matrixData
size_t transformSizeInBytes = sizeof(OptixMatrixMotionTransform) + (N-2) * 12 * sizeof(float);
OptixMatrixMotionTransform* 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() This struct is interpreted on the device by rtcore, and should mirror RtcTravMatrixMotionTransform.

7.38.2 Member Data Documentation

7.38.2.1 child

`OptixTraversableHandle OptixMatrixMotionTransform::child`

The traversable that is transformed by this transformation.

7.38.2.2 motionOptions

`OptixMotionOptions OptixMatrixMotionTransform::motionOptions`

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

7.38.2.3 pad

`unsigned int OptixMatrixMotionTransform::pad[3]`

Padding to make the transformation 16 byte aligned.

7.38.2.4 transform

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

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

7.39 OptixMicromapBuffers Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.39.1 Detailed Description

Buffer inputs for opacity micromap array builds.

7.39.2 Member Data Documentation

7.39.2.1 output

`CUdeviceptr OptixMicromapBuffers::output`

Output buffer.

7.39.2.2 outputSizeInBytes

`size_t OptixMicromapBuffers::outputSizeInBytes`

Output buffer size.

7.39.2.3 temp

`CUdeviceptr OptixMicromapBuffers::temp`

Temp buffer.

7.39.2.4 tempSizeInBytes

`size_t OptixMicromapBuffers::tempSizeInBytes`

Temp buffer size.

7.40 OptixMicromapBufferSizes Struct Reference

`#include <optix_types.h>`

Public Attributes

- size_t outputSizeInBytes
- size_t tempSizeInBytes

7.40.1 Detailed Description

Conservative memory requirements for building a opacity micromap array.

7.40.2 Member Data Documentation

7.40.2.1 outputSizeInBytes

`size_t OptixMicromapBufferSizes::outputSizeInBytes`

7.40.2.2 tempSizeInBytes

```
size_t OptixMicromapBufferSizes::tempSizeInBytes
```

7.41 OptixModuleCompileBoundValueEntry Struct Reference

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

Public Attributes

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

7.41.1 Detailed Description

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

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

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

7.41.2 Member Data Documentation

7.41.2.1 annotation

```
const char* OptixModuleCompileBoundValueEntry::annotation
```

7.41.2.2 boundValuePtr

```
const void* OptixModuleCompileBoundValueEntry::boundValuePtr
```

7.41.2.3 pipelineParamOffsetInBytes

```
size_t OptixModuleCompileBoundValueEntry::pipelineParamOffsetInBytes
```

7.41.2.4 sizeInBytes

```
size_t OptixModuleCompileBoundValueEntry::sizeInBytes
```

7.42 OptixModuleCompileOptions Struct Reference

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

Public Attributes

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

7.42.1 Detailed Description

Compilation options for module.

See also optixModuleCreate( )

7.42.2 Member Data Documentation

7.42.2.1 boundValues

```
const OptixModuleCompileBoundValueEntry* OptixModuleCompileOptions
::boundValues
```

Ingored if numBoundValues is set to 0.

7.42.2.2 debugLevel

```
OptixCompileDebugLevel OptixModuleCompileOptions::debugLevel
```

Generate debug information.

7.42.2.3 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.

7.42.2.4 numBoundValues

```
unsigned int OptixModuleCompileOptions::numBoundValues
```

set to 0 if unused

7.42.2.5 numPayloadTypes

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

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

7.42.2.6 optLevel

`OptixCompileOptimizationLevel OptixModuleCompileOptions::optLevel`

Optimization level. May vary within a pipeline.

7.42.2.7 payloadTypes

`const OptixPayloadType* OptixModuleCompileOptions::payloadTypes`

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

7.43 OptixMotionOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.43.1 Detailed Description

Motion options.

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

7.43.2 Member Data Documentation

7.43.2.1 flags

`unsigned short OptixMotionOptions::flags`

Combinations of OptixMotionFlags.

7.43.2.2 numKeys

`unsigned short OptixMotionOptions::numKeys`

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

7.43.2.3 timeBegin

`float OptixMotionOptions::timeBegin`

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

7.43.2.4 timeEnd

`float OptixMotionOptions::timeEnd`

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

7.44 OptixNetworkDescription Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixCoopVecMatrixDescription * layers
- unsigned int numLayers

7.44.1 Member Data Documentation

7.44.1.1 layers

```
OptixCoopVecMatrixDescription* OptixNetworkDescription::layers
```

7.44.1.2 numLayers

```
unsigned int OptixNetworkDescription::numLayers
```

7.45 OptixOpacityMicromapArrayBuildInput Struct Reference

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

Public Attributes

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

7.45.1 Detailed Description

Inputs to opacity micromap array construction.

7.45.2 Member Data Documentation

7.45.2.1 flags

```
unsigned int OptixOpacityMicromapArrayBuildInput::flags
```

Applies to all opacity micromaps in array.

7.45.2.2 inputBuffer

```
CUdeviceptr OptixOpacityMicromapArrayBuildInput::inputBuffer
```

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

7.45.2.3 micromapHistogramEntries

```
const OptixOpacityMicromapHistogramEntry*
OptixOpacityMicromapArrayBuildInput::micromapHistogramEntries
```

Histogram over opacity micromaps of input format and subdivision combinations. Counts of entries with equal format and subdivision combination (duplicates) are added together.

7.45.2.4 numMicromapHistogramEntries

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

Number of OptixOpacityMicromapHistogramEntry.

7.45.2.5 perMicromapDescBuffer

```
CUdeviceptr OptixOpacityMicromapArrayBuildInput::perMicromapDescBuffer
```

One OptixOpacityMicromapDesc entry per opacity micromap. This device pointer must be a multiple of OPTIX_OPACITY_MICROMAP_DESC_BYTE_ALIGNMENT.

7.45.2.6 perMicromapDescStrideInBytes

```
unsigned int OptixOpacityMicromapArrayBuildInput
::perMicromapDescStrideInBytes
```

Stride between OptixOpacityMicromapDescs in perOmDescBuffer. If set to zero, the opacity micromap descriptors are assumed to be tightly packed and the stride is assumed to be sizeof(OptixOpacityMicromapDesc). This stride must be a multiple of OPTIX_OPACITY_MICROMAP_DESC_BYTE_ALIGNMENT.

7.46 OptixOpacityMicromapDesc Struct Reference

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

Public Attributes

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

7.46.1 Detailed Description

Opacity micromap descriptor.

7.46.2 Member Data Documentation

7.46.2.1 byteOffset

```
unsigned int OptixOpacityMicromapDesc::byteOffset
```

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

7.46.2.2 format

```
unsigned short OptixOpacityMicromapDesc::format
```

OptixOpacityMicromapFormat.

7.46.2.3 subdivisionLevel

```
unsigned short OptixOpacityMicromapDesc::subdivisionLevel
```

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

7.47 OptixOpacityMicromapHistogramEntry Struct Reference

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

Public Attributes

- unsigned int count
- unsigned int subdivisionLevel
- OptixOpacityMicromapFormat format

7.47.1 Detailed Description

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

7.47.2 Member Data Documentation

7.47.2.1 count

```
unsigned int OptixOpacityMicromapHistogramEntry::count
```

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

7.47.2.2 format

```
OptixOpacityMicromapFormat OptixOpacityMicromapHistogramEntry::format
```

Opacity micromap format.

7.47.2.3 subdivisionLevel

```
unsigned int OptixOpacityMicromapHistogramEntry::subdivisionLevel
```

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

7.48 OptixOpacityMicromapUsageCount Struct Reference

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

Public Attributes

- unsigned int count
- unsigned int subdivisionLevel
- OptixOpacityMicromapFormat format

7.48.1 Detailed Description

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

7.48.2 Member Data Documentation

7.48.2.1 count

```
unsigned int OptixOpacityMicromapUsageCount::count
```

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

7.48.2.2 format

`OptixOpacityMicromapFormat OptixOpacityMicromapUsageCount::format`

opacity micromap format.

7.48.2.3 subdivisionLevel

`unsigned int OptixOpacityMicromapUsageCount::subdivisionLevel`

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

7.49 OptixOutgoingHitObject Struct Reference

`#include <optix_device.h>`

Public Member Functions

- __forceinline__ __device__ float getRayTime () const
- __forceinline__ __device__ unsigned int getTransformListSize () const
- __forceinline__ __device__ OptixTraversableHandle getTransformListHandle (unsigned int index) const

7.49.1 Member Function Documentation

7.49.1.1 getRayTime()

```
__forceinline__ __device__ float OptixOutgoingHitObject::getRayTime ( )
const  [inline]
```

7.49.1.2 getTransformListHandle()

```
__forceinline__ __device__ OptixTraversableHandle OptixOutgoingHitObject
::getTransformListHandle (
          unsigned int index ) const  [inline]
```

7.49.1.3 getTransformListSize()

```
__forceinline__ __device__ unsigned int OptixOutgoingHitObject
::getTransformListSize ( ) const  [inline]
```

7.50 OptixPayloadType Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int numPayloadValues
- const unsigned int * payloadSemantics

7.50.1 Detailed Description

Specifies a single payload type.

7.50.2 Member Data Documentation

7.50.2.1 numPayloadValues

`unsigned int OptixPayloadType::numPayloadValues`

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

7.50.2.2 payloadSemantics

`const unsigned int* OptixPayloadType::payloadSemantics`

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

7.51 OptixPipelineCompileOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

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

7.51.1 Detailed Description

Compilation options for all modules of a pipeline.

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

See also optixModuleCreate( ), optixPipelineCreate( )

7.51.2 Member Data Documentation

7.51.2.1 allowClusteredGeometry

`int OptixPipelineCompileOptions::allowClusteredGeometry`

Boolean value indicating whether clusters (cluster acceleration structure) may used.

7.51.2.2 allowOpacityMicromaps

`int OptixPipelineCompileOptions::allowOpacityMicromaps`

Boolean value indicating whether opacity micromaps may used.

7.51.2.3 exceptionFlags

`unsigned int OptixPipelineCompileOptions::exceptionFlags`

A bitmask of OptixExceptionFlags indicating which exceptions are enabled.

7.51.2.4 numAttributeValues

`int OptixPipelineCompileOptions::numAttributeValues`

How much storage, in 32b words, to make available for the attributes. The minimum number is 2. Values below that will automatically be changed to 2. [2..8].

7.51.2.5 numPayloadValues

`int OptixPipelineCompileOptions::numPayloadValues`

How much storage, in 32b words, to make available for the payload, [0..32] Must be zero if numPayloadTypes is not zero.

7.51.2.6 pipelineLaunchParamsVariableName

`const char* OptixPipelineCompileOptions::pipelineLaunchParamsVariableName`

The name of the pipeline parameter variable. If 0, no pipeline parameter will be available. This will be ignored if the launch param variable was optimized out or was not found in the modules linked to the pipeline.

7.51.2.7 traversableGraphFlags

`unsigned int OptixPipelineCompileOptions::traversableGraphFlags`

Traversable graph bitfield. See OptixTraversableGraphFlags.

7.51.2.8 usesMotionBlur

`int OptixPipelineCompileOptions::usesMotionBlur`

Boolean value indicating whether motion blur could be used.

7.51.2.9 usesPrimitiveTypeFlags

`unsigned int OptixPipelineCompileOptions::usesPrimitiveTypeFlags`

Bit field enabling primitive types. See OptixPrimitiveTypeFlags. Setting to zero corresponds to enabling OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM and OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE.

7.52 OptixPipelineLinkOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int maxTraceDepth

7.52.1 Detailed Description

Link options for a pipeline.

See also optixPipelineCreate( )

7.52.2 Member Data Documentation

7.52.2.1 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.

7.53 OptixProgramGroupCallables Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixModule moduleDC
- const char * entryFunctionNameDC
- OptixModule moduleCC
- const char * entryFunctionNameCC

7.53.1 Detailed Description

Program group representing callables.

Module and entry function name need to be valid for at least one of the two callables.

See also #OptixProgramGroupDesc::callables

7.53.2 Member Data Documentation

7.53.2.1 entryFunctionNameCC

`const char* OptixProgramGroupCallables::entryFunctionNameCC`

Entry function name of the continuation callable (CC) program.

7.53.2.2 entryFunctionNameDC

`const char* OptixProgramGroupCallables::entryFunctionNameDC`

Entry function name of the direct callable (DC) program.

7.53.2.3 moduleCC

`OptixModule OptixProgramGroupCallables::moduleCC`

Module holding the continuation callable (CC) program.

7.53.2.4 moduleDC

`OptixModule OptixProgramGroupCallables::moduleDC`

Module holding the direct callable (DC) program.

7.54 OptixProgramGroupDesc Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixProgramGroupKind kind

- unsigned int flags
- union {
  OptixProgramGroupSingleModule raygen
  OptixProgramGroupSingleModule miss
  OptixProgramGroupSingleModule exception
  OptixProgramGroupCallables callables
  OptixProgramGroupHitgroup hitgroup
  };

7.54.1 Detailed Description

Descriptor for program groups.

7.54.2 Member Data Documentation

7.54.2.1

`union { ... } OptixProgramGroupDesc::@9`

7.54.2.2 callables

`OptixProgramGroupCallables OptixProgramGroupDesc::callables`

See also OPTIX_PROGRAM_GROUP_KIND_CALLABLES

7.54.2.3 exception

`OptixProgramGroupSingleModule OptixProgramGroupDesc::exception`

See also OPTIX_PROGRAM_GROUP_KIND_EXCEPTION

7.54.2.4 flags

`unsigned int OptixProgramGroupDesc::flags`

See OptixProgramGroupFlags.

7.54.2.5 hitgroup

`OptixProgramGroupHitgroup OptixProgramGroupDesc::hitgroup`

See also OPTIX_PROGRAM_GROUP_KIND_HITGROUP

7.54.2.6 kind

`OptixProgramGroupKind OptixProgramGroupDesc::kind`

The kind of program group.

7.54.2.7 miss

`OptixProgramGroupSingleModule OptixProgramGroupDesc::miss`

See also OPTIX_PROGRAM_GROUP_KIND_MISS

7.54.2.8 raygen

`OptixProgramGroupSingleModule OptixProgramGroupDesc::raygen`

See also OPTIX_PROGRAM_GROUP_KIND_RAYGEN

7.55 OptixProgramGroupHitgroup Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixModule moduleCH
- const char * entryFunctionNameCH
- OptixModule moduleAH
- const char * entryFunctionNameAH
- OptixModule moduleIS
- const char * entryFunctionNameIS

7.55.1 Detailed Description

Program group representing the hitgroup.

For each of the three program types, module and entry function name might both be `nullptr`.

See also OptixProgramGroupDesc::hitgroup

7.55.2 Member Data Documentation

7.55.2.1 entryFunctionNameAH

`const char* OptixProgramGroupHitgroup::entryFunctionNameAH`

Entry function name of the any hit (AH) program.

7.55.2.2 entryFunctionNameCH

`const char* OptixProgramGroupHitgroup::entryFunctionNameCH`

Entry function name of the closest hit (CH) program.

7.55.2.3 entryFunctionNameIS

`const char* OptixProgramGroupHitgroup::entryFunctionNameIS`

Entry function name of the intersection (IS) program.

7.55.2.4 moduleAH

`OptixModule OptixProgramGroupHitgroup::moduleAH`

Module holding the any hit (AH) program.

7.55.2.5 moduleCH

`OptixModule OptixProgramGroupHitgroup::moduleCH`

Module holding the closest hit (CH) program.

7.55.2.6 moduleIS

`OptixModule OptixProgramGroupHitgroup::moduleIS`

Module holding the intersection (Is) program.

7.56 OptixProgramGroupOptions Struct Reference

`#include <optix_types.h>`

Public Attributes

- const OptixPayloadType * payloadType

7.56.1 Detailed Description

Program group options.

See also optixProgramGroupCreate( )

7.56.2 Member Data Documentation

7.56.2.1 payloadType

`const OptixPayloadType* OptixProgramGroupOptions::payloadType`

Specifies the payload type of this program group. All programs in the group must support the payload type (Program support for a type is specified by calling.

See also optixSetPayloadTypes or otherwise all types specified in

OptixModuleCompileOptions are supported). If a program is not available for the requested payload type, optixProgramGroupCreate returns OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH. If the payloadType is left zero, a unique type is deduced. The payload type can be uniquely deduced if there is exactly one payload type for which all programs in the group are available. If the payload type could not be deduced uniquely optixProgramGroupCreate returns OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED.

7.57 OptixProgramGroupSingleModule Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixModule module
- const char * entryFunctionName

7.57.1 Detailed Description

Program group representing a single module.

Used for raygen, miss, and exception programs. In case of raygen and exception programs, module and entry function name need to be valid. For miss programs, module and entry function name might both be `nullptr`.

See also OptixProgramGroupDesc::raygen, OptixProgramGroupDesc::miss, OptixProgramGroupDesc::exception

7.57.2 Member Data Documentation

7.57.2.1 entryFunctionName

`const char* OptixProgramGroupSingleModule::entryFunctionName`

Entry function name of the single program.

7.57.2.2 module

`OptixModule OptixProgramGroupSingleModule::module`

Module holding single program.

7.58 OptixRelocateInput Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixBuildInputType type
- union {
  OptixRelocateInputInstanceArray instanceArray
  OptixRelocateInputTriangleArray triangleArray
  };

7.58.1 Detailed Description

Relocation inputs.

See also optixAccelRelocate( )

7.58.2 Member Data Documentation

7.58.2.1

`union { ... } OptixRelocateInput::@3`

7.58.2.2 instanceArray

`OptixRelocateInputInstanceArray OptixRelocateInput::instanceArray`

Instance and instance pointer inputs.

7.58.2.3 triangleArray

`OptixRelocateInputTriangleArray OptixRelocateInput::triangleArray`

Triangle inputs.

7.58.2.4 type

`OptixBuildInputType OptixRelocateInput::type`

The type of the build input to relocate.

7.59 OptixRelocateInputInstanceArray Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int numInstances
- CUdeviceptr traversableHandles

7.59.1 Detailed Description

Instance and instance pointer inputs.

See also OptixRelocateInput::instanceArray

7.59.2 Member Data Documentation

7.59.2.1 numInstances

```
unsigned int OptixRelocateInputInstanceArray::numInstances
```

Number of elements in OptixRelocateInputInstanceArray::traversableHandles. Must match OptixBuildInputInstanceArray::numInstances of the source build input.

7.59.2.2 traversableHandles

```
CUdeviceptr OptixRelocateInputInstanceArray::traversableHandles
```

These are the traversable handles of the instances (See OptixInstance::traversableHandle) These can be used when also relocating the instances. No updates to the bounds are performed. Use optixAccelBuild to update the bounds. 'traversableHandles' may be zero when the traversables are not relocated (i.e. relocation of an IAS on the source device).

7.60 OptixRelocateInputOpacityMicromap Struct Reference

```
#include <optix_types.h>
```

Public Attributes

- CUdeviceptr opacityMicromapArray

7.60.1 Member Data Documentation

7.60.1.1 opacityMicromapArray

```
CUdeviceptr OptixRelocateInputOpacityMicromap::opacityMicromapArray
```

Device pointer to a relocated opacity micromap array used by the source build input array. May be zero when no micromaps 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).

7.61 OptixRelocateInputTriangleArray Struct Reference

```
#include <optix_types.h>
```

Public Attributes

- unsigned int numSbtRecords
- OptixRelocateInputOpacityMicromap opacityMicromap

7.61.1 Detailed Description

Triangle inputs.

See also OptixRelocateInput::triangleArray

7.61.2 Member Data Documentation

7.61.2.1 numSbtRecords

```
unsigned int OptixRelocateInputTriangleArray::numSbtRecords
```

Number of sbt records available to the sbt index offset override. Must match OptixBuildInputTriangleArray::numSbtRecords of the source build input.

7.61.2.2 opacityMicromap

```
OptixRelocateInputOpacityMicromap OptixRelocateInputTriangleArray
::opacityMicromap
```

Opacity micromap inputs.

7.62 OptixRelocationInfo Struct Reference

```
#include <optix_types.h>
```

Public Attributes

- unsigned long long info [4]

7.62.1 Detailed Description

Used to store information related to relocation of optix data structures.

See also optixOpacityMicromapArrayGetRelocationInfo( ), optixOpacityMicromapArrayRelocate( ), optixAccelGetRelocationInfo( ), optixAccelRelocate( ), optixCheckRelocationCompatibility( )

7.62.2 Member Data Documentation

7.62.2.1 info

```
unsigned long long OptixRelocationInfo::info[4]
```

Opaque data, used internally, should not be modified.

7.63 OptixShaderBindingTable Struct Reference

```
#include <optix_types.h>
```

Public Attributes

- CUdeviceptr raygenRecord
- CUdeviceptr exceptionRecord

- CUdeviceptr missRecordBase
- unsigned int missRecordStrideInBytes
- unsigned int missRecordCount

- CUdeviceptr hitgroupRecordBase
- unsigned int hitgroupRecordStrideInBytes
- unsigned int hitgroupRecordCount

- CUdeviceptr callablesRecordBase
- unsigned int callablesRecordStrideInBytes
- unsigned int callablesRecordCount

7.63.1 Detailed Description

Describes the shader binding table (SBT)

See also optixLaunch( )

7.63.2 Member Data Documentation

7.63.2.1 callablesRecordBase

```
CUdeviceptr OptixShaderBindingTable::callablesRecordBase
```

Arrays of SBT records for callable programs. If the base address is not null, the stride and count must not be zero. If the base address is null, then the count needs to zero. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.63.2.2 callablesRecordCount

```
unsigned int OptixShaderBindingTable::callablesRecordCount
```

Arrays of SBT records for callable programs. If the base address is not null, the stride and count must not be zero. If the base address is null, then the count needs to zero. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.63.2.3 callablesRecordStrideInBytes

```
unsigned int OptixShaderBindingTable::callablesRecordStrideInBytes
```

Arrays of SBT records for callable programs. If the base address is not null, the stride and count must not be zero. If the base address is null, then the count needs to zero. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.63.2.4 exceptionRecord

```
CUdeviceptr OptixShaderBindingTable::exceptionRecord
```

Device address of the SBT record of the exception program. The address must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.63.2.5 hitgroupRecordBase

```
CUdeviceptr OptixShaderBindingTable::hitgroupRecordBase
```

Arrays of SBT records for hit groups. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.63.2.6 hitgroupRecordCount

```
unsigned int OptixShaderBindingTable::hitgroupRecordCount
```

Arrays of SBT records for hit groups. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.63.2.7 hitgroupRecordStrideInBytes

```
unsigned int OptixShaderBindingTable::hitgroupRecordStrideInBytes
```

Arrays of SBT records for hit groups. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.63.2.8 missRecordBase

`CUdeviceptr OptixShaderBindingTable::missRecordBase`

Arrays of SBT records for miss programs. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.63.2.9 missRecordCount

`unsigned int OptixShaderBindingTable::missRecordCount`

Arrays of SBT records for miss programs. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.63.2.10 missRecordStrideInBytes

`unsigned int OptixShaderBindingTable::missRecordStrideInBytes`

Arrays of SBT records for miss programs. The base address and the stride must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.63.2.11 raygenRecord

`CUdeviceptr OptixShaderBindingTable::raygenRecord`

Device address of the SBT record of the ray gen program to start launch at. The address must be a multiple of OPTIX_SBT_RECORD_ALIGNMENT.

7.64 OptixSRTData Struct Reference

`#include <optix_types.h>`

Public Attributes

Parameters describing the SRT transformation

- float sx
- float a
- float b
- float pvx
- float sy
- float c
- float pvy
- float sz
- float pvz
- float qx
- float qy
- float qz
- float qw
- float tx
- float ty
- float tz

7.64.1 Detailed Description

Represents an SRT transformation.

An SRT transformation can represent a smooth rotation with fewer motion keys than a matrix transformation. Each motion key is constructed from elements taken from a matrix S, a quaternion R, and a translation T.

The scaling matrix $S = \begin{bmatrix} sx & a & b & pvx \\ 0 & sy & c & pvy \\ 0 & 0 & sz & pvz \end{bmatrix}$ defines an affine transformation that can include scale, shear, and a translation. The translation allows to define the pivot point for the subsequent rotation.

The quaternion R = [qx, qy, qz, qw] describes a rotation with angular component qw = cos(theta/2) and other components [qx, qy, qz] = sin(theta/2) $*$ [ax, ay, az] where the axis [ax, ay, az] is normalized.

The translation matrix $T = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \end{bmatrix}$ defines another translation that is applied after the rotation. Typically, this translation includes the inverse translation from the matrix S to reverse the translation for the pivot point for R.

To obtain the effective transformation at time t, the elements of the components of S, R, and T will be interpolated linearly. The components are then multiplied to obtain the combined transformation C = T $*$ R $*$ S. The transformation C is the effective object-to-world transformations at time t, and C^(-1) is the effective world-to-object transformation at time t.

See also OptixSRTMotionTransform::srtData, optixConvertPointerToTraversableHandle()

7.64.2 Member Data Documentation

7.64.2.1 a

```
float OptixSRTData::a
```

7.64.2.2 b

```
float OptixSRTData::b
```

7.64.2.3 c

```
float OptixSRTData::c
```

7.64.2.4 pvx

```
float OptixSRTData::pvx
```

7.64.2.5 pvy

```
float OptixSRTData::pvy
```

7.64.2.6 pvz

```
float OptixSRTData::pvz
```

7.64.2.7 qw

```
float OptixSRTData::qw
```

7.64.2.8 qx

```
float OptixSRTData::qx
```

7.64.2.9 qy

`float OptixSRTData::qy`

7.64.2.10 qz

`float OptixSRTData::qz`

7.64.2.11 sx

`float OptixSRTData::sx`

7.64.2.12 sy

`float OptixSRTData::sy`

7.64.2.13 sz

`float OptixSRTData::sz`

7.64.2.14 tx

`float OptixSRTData::tx`

7.64.2.15 ty

`float OptixSRTData::ty`

7.64.2.16 tz

`float OptixSRTData::tz`

7.65 OptixSRTMotionTransform Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixTraversableHandle child
- OptixMotionOptions motionOptions
- unsigned int pad [3]
- OptixSRTData srtData [2]

7.65.1 Detailed Description

Represents an SRT motion transformation.

The device address of instances of this type must be a multiple of OPTIX_TRANSFORM_BYTE_ALIGNMENT.

This struct, as defined here, handles only N=2 motion keys due to the fixed array length of its srtData member. The following example shows how to create instances for an arbitrary number N of motion keys:

```
OptixSRTData srtData[N];
... // setup srtData
size_t transformSizeInBytes = sizeof(OptixSRTMotionTransform) + (N-2) * sizeof(OptixSRTData);
OptixSRTMotionTransform* srtMotionTransform = (OptixSRTMotionTransform*) malloc(transformSizeInBytes);
memset(srtMotionTransform, 0, transformSizeInBytes);
... // setup other members of srtMotionTransform
```

```
srtMotionTransform->motionOptions.numKeys   = N;
memcpy(srtMotionTransform->srtData, srtData, N * sizeof(OptixSRTData));
... // copy srtMotionTransform to device memory
free(srtMotionTransform)
```

See also optixConvertPointerToTraversableHandle() This struct is interpreted on the device by rtcore, and should mirror RtcTravSRTMotionTransform.

7.65.2 Member Data Documentation

7.65.2.1 child

`OptixTraversableHandle OptixSRTMotionTransform::child`

The traversable transformed by this transformation.

7.65.2.2 motionOptions

`OptixMotionOptions OptixSRTMotionTransform::motionOptions`

The motion options for this transformation Must have at least two motion keys.

7.65.2.3 pad

`unsigned int OptixSRTMotionTransform::pad[3]`

Padding to make the SRT data 16 byte aligned.

7.65.2.4 srtData

`OptixSRTData OptixSRTMotionTransform::srtData[2]`

The actual SRT data describing the transformation.

7.66 OptixStackSizes Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int cssRG
- unsigned int cssMS
- unsigned int cssCH
- unsigned int cssAH
- unsigned int cssIS
- unsigned int cssCC
- unsigned int dssDC

7.66.1 Detailed Description

Describes the stack size requirements of a program group.

See also optixProgramGroupGetStackSize()

7.66.2 Member Data Documentation

7.66.2.1 cssAH

`unsigned int OptixStackSizes::cssAH`

Continuation stack size of AH programs in bytes.

7.66.2.2 cssCC

`unsigned int OptixStackSizes::cssCC`

Continuation stack size of CC programs in bytes.

7.66.2.3 cssCH

`unsigned int OptixStackSizes::cssCH`

Continuation stack size of CH programs in bytes.

7.66.2.4 cssIS

`unsigned int OptixStackSizes::cssIS`

Continuation stack size of IS programs in bytes.

7.66.2.5 cssMS

`unsigned int OptixStackSizes::cssMS`

Continuation stack size of MS programs in bytes.

7.66.2.6 cssRG

`unsigned int OptixStackSizes::cssRG`

Continuation stack size of RG programs in bytes.

7.66.2.7 dssDC

`unsigned int OptixStackSizes::dssDC`

Direct stack size of DC programs in bytes.

7.67 OptixStaticTransform Struct Reference

`#include <optix_types.h>`

Public Attributes

- OptixTraversableHandle child
- unsigned int pad [2]
- float transform [12]
- float invTransform [12]

7.67.1 Detailed Description

Static transform.

The device address of instances of this type must be a multiple of OPTIX_TRANSFORM_BYTE_ALIGNMENT.

See also optixConvertPointerToTraversableHandle( ) This struct is interpreted on the device by rtcore, and should mirror RtcTravStaticTransform.

7.67.2 Member Data Documentation

7.67.2.1 child

`OptixTraversableHandle OptixStaticTransform::child`

The traversable transformed by this transformation.

7.67.2.2 invTransform

`float OptixStaticTransform::invTransform[12]`

Affine world-to-object transformation as 3x4 matrix in row-major layout Must be the inverse of the transform matrix.

7.67.2.3 pad

`unsigned int OptixStaticTransform::pad[2]`

Padding to make the transformations 16 byte aligned.

7.67.2.4 transform

`float OptixStaticTransform::transform[12]`

Affine object-to-world transformation as 3x4 matrix in row-major layout.

7.68 OptixTraverseData Struct Reference

`#include <optix_types.h>`

Public Attributes

- unsigned int data [20]

7.68.1 Detailed Description

Hit Object Struct to store the data collected in a hit object during traversal in an internal format using `optixHitObjectGetTraverseData( )`. The hit object can be reconstructed using that data at a later point with `optixMakeHitObjectWithTraverseData( )`.

7.68.2 Member Data Documentation

7.68.2.1 data

`unsigned int OptixTraverseData::data[20]`

7.69 OptixUtilDenoiserImageTile Struct Reference

`#include <optix_denoiser_tiling.h>`

Public Attributes

- OptixImage2D input
- OptixImage2D output
- unsigned int inputOffsetX
- unsigned int inputOffsetY

7.69.1 Detailed Description

Tile definition.

see optixUtilDenoiserSplitImage

7.69.2 Member Data Documentation

7.69.2.1 input

```
OptixImage2D OptixUtilDenoiserImageTile::input
```

7.69.2.2 inputOffsetX

```
unsigned int OptixUtilDenoiserImageTile::inputOffsetX
```

7.69.2.3 inputOffsetY

```
unsigned int OptixUtilDenoiserImageTile::inputOffsetY
```

7.69.2.4 output

```
OptixImage2D OptixUtilDenoiserImageTile::output
```

7.70 optix_internal::TypePack<... > Struct Template Reference

```
#include <optix_device_impl.h>
```

8 File Documentation

8.1 optix_device_impl.h File Reference

Classes

- struct optix_internal::TypePack<... >

Namespaces

- namespace optix_internal

Macros

- #define OPTIX_DEFINE_optixGetAttribute_BODY(which)
- #define OPTIX_DEFINE_optixGetExceptionDetail_BODY(which)

Functions

- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTraverse (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)

- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTraverse (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- static __forceinline__ __device__ void optixReorder (unsigned int coherenceHint, unsigned int numCoherenceHintBits)
- static __forceinline__ __device__ void optixReorder ()
- template<typename... Payload>
  static __forceinline__ __device__ void optixInvoke (OptixPayloadTypeID type, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixInvoke (Payload &... payload)
- static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float rayTime, unsigned int rayFlags, OptixTraverseData traverseData, const OptixTraversableHandle *transforms, unsigned int numTransforms)
- static __forceinline__ __device__ void optixMakeMissHitObject (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int rayFlags)
- static __forceinline__ __device__ void optixMakeNopHitObject ()
- static __forceinline__ __device__ void optixHitObjectGetTraverseData (OptixTraverseData *data)
- static __forceinline__ __device__ bool optixHitObjectIsHit ()
- static __forceinline__ __device__ bool optixHitObjectIsMiss ()
- static __forceinline__ __device__ bool optixHitObjectIsNop ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize ()
- static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmin ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmax ()
- static __forceinline__ __device__ float optixHitObjectGetRayTime ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex ()

- static __forceinline__ __device__ void optixHitObjectSetSbtRecordIndex (unsigned int sbtRecordIndex)
- static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer ()
- static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetGASTraversableHandle ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetRayFlags ()
- static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_5 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_6 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_7 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_8 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_9 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_10 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_11 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_12 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_13 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_14 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_15 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_16 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_17 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_18 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_19 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_20 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_21 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_22 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_23 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_24 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_25 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_26 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_27 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_28 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_29 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_30 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_31 (unsigned int p)
- static __forceinline__ __device__ unsigned int optixGetPayload_0 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_1 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_2 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_3 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_4 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_5 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_6 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_7 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_8 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_9 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_10 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_11 ()

- static __forceinline__ __device__ unsigned int optixGetPayload_12 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_13 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_14 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_15 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_16 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_17 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_18 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_19 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_20 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_21 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_22 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_23 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_24 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_25 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_26 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_27 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_28 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_29 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_30 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_31 ()
- static __forceinline__ __device__ void optixSetPayloadTypes (unsigned int types)
- static __forceinline__ __device__ unsigned int optixUndefinedValue ()
- static __forceinline__ __device__ float3 optixGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixGetObjectRayOrigin ()
- static __forceinline__ __device__ float3 optixGetObjectRayDirection ()
- static __forceinline__ __device__ float optixGetRayTmin ()
- static __forceinline__ __device__ float optixGetRayTmax ()
- static __forceinline__ __device__ float optixGetRayTime ()
- static __forceinline__ __device__ unsigned int optixGetRayFlags ()
- static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceTraversableFromIAS (OptixTraversableHandle ias, unsigned int instIdx)
- static __forceinline__ __device__ void optixGetTriangleVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static __forceinline__ __device__ void optixGetTriangleVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static __forceinline__ __device__ void optixGetTriangleVertexData (float3 data[3])
- static __forceinline__ __device__ void optixHitObjectGetTriangleVertexData (float3 data[3])
- static __forceinline__ __device__ void optixGetLinearCurveVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void optixGetLinearCurveVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void optixGetLinearCurveVertexData (float4 data[2])
- static __forceinline__ __device__ void optixHitObjectGetLinearCurveVertexData (float4 data[2])
- static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])

- static __forceinline__ __device__ void optixGetQuadraticBSplineVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (float4 data[3])
- static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineVertexData (float4 data[3])
- static __forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexData (float4 data[3])
- static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineRocapsVertexData (float4 data[3])
- static __forceinline__ __device__ void optixGetCubicBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBSplineVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBSplineVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCubicBSplineVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCubicBSplineRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCatmullRomVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCatmullRomRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCubicBezierVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCubicBezierRocapsVertexData (float4 data[4])

- static __forceinline__ __device__ void optixGetRibbonVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetRibbonVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetRibbonVertexData (float4 data[3])
- static __forceinline__ __device__ void optixHitObjectGetRibbonVertexData (float4 data[3])
- static __forceinline__ __device__ float3 optixGetRibbonNormal (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ float3 optixGetRibbonNormalFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ float3 optixGetRibbonNormal (float2 ribbonParameters)
- static __forceinline__ __device__ float3 optixHitObjectGetRibbonNormal (float2 ribbonParameters)
- static __forceinline__ __device__ void optixGetSphereData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ void optixGetSphereDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ void optixGetSphereData (float4 data[1])
- static __forceinline__ __device__ void optixHitObjectGetSphereData (float4 data[1])
- static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle ()
- static __forceinline__ __device__ float optixGetGASMotionTimeBegin (OptixTraversableHandle handle)
- static __forceinline__ __device__ float optixGetGASMotionTimeEnd (OptixTraversableHandle handle)
- static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (OptixTraversableHandle handle)
- template<typename HitState >
  static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (const HitState &hs, float m[12])
- static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixHitObjectGetWorldToObjectTransformMatrix (float m[12])
- template<typename HitState >
  static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (const HitState &hs, float m[12])
- static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixHitObjectGetObjectToWorldTransformMatrix (float m[12])
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (const HitState &hs, float3 point)
- static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (float3 point)
- static __forceinline__ __device__ float3 optixHitObjectTransformPointFromWorldToObjectSpace (float3 point)
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (const HitState &hs, float3 vec)
- static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (float3 vec)

- static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromWorldToObjectSpace (float3 vec)
- template<typename HitState > static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (const HitState &hs, float3 normal)
- static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (float3 normal)
- static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromWorldToObjectSpace (float3 normal)
- template<typename HitState > static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (const HitState &hs, float3 point)
- static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (float3 point)
- static __forceinline__ __device__ float3 optixHitObjectTransformPointFromObjectToWorldSpace (float3 point)
- template<typename HitState > static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (const HitState &hs, float3 vec)
- static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (float3 vec)
- static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromObjectToWorldSpace (float3 vec)
- template<typename HitState > static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (const HitState &hs, float3 normal)
- static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (float3 normal)
- static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromObjectToWorldSpace (float3 normal)
- static __forceinline__ __device__ unsigned int optixGetTransformListSize ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTransformType optixGetTransformTypeFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixStaticTransform * optixGetStaticTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixSRTMotionTransform * optixGetSRTMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixMatrixMotionTransform * optixGetMatrixMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceChildFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceInverseTransformFromHandle (OptixTraversableHandle handle)
- static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind)

- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6, unsigned int a7)
- static __forceinline__ __device__ unsigned int optixGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_7 ()
- static __forceinline__ __device__ void optixTerminateRay ()
- static __forceinline__ __device__ void optixIgnoreIntersection ()
- static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixGetClusterId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetClusterId ()
- static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixGetHitKind ()
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsBackFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsFrontFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ()
- static __forceinline__ __device__ bool optixIsBackFaceHit ()
- static __forceinline__ __device__ bool optixIsFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleHit ()
- static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ()
- static __forceinline__ __device__ float optixGetCurveParameter ()
- static __forceinline__ __device__ float optixHitObjectGetCurveParameter ()
- static __forceinline__ __device__ float2 optixGetRibbonParameters ()
- static __forceinline__ __device__ float2 optixHitObjectGetRibbonParameters ()
- static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ()

- static __forceinline__ __device__ float2 optixHitObjectGetTriangleBarycentrics ()
- static __forceinline__ __device__ uint3 optixGetLaunchIndex ()
- static __forceinline__ __device__ uint3 optixGetLaunchDimensions ()
- static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ()
- static __forceinline__ __device__ void optixThrowException (int exceptionCode)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6, unsigned int exceptionDetail7)
- static __forceinline__ __device__ int optixGetExceptionCode ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7 ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable ()
- static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset ()
- static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay ()
- static __forceinline__ __device__ OptixParameterMismatchExceptionDetails optixGetExceptionParameterMismatch ()
- static __forceinline__ __device__ char * optixGetExceptionLineInfo ()
- template<typename ReturnT , typename... ArgTypes>
  static __forceinline__ __device__ ReturnT optixDirectCall (unsigned int sbtIndex, ArgTypes... args)
- template<typename ReturnT , typename... ArgTypes>
  static __forceinline__ __device__ ReturnT optixContinuationCall (unsigned int sbtIndex, ArgTypes... args)

- static __forceinline__ __device__ uint4 optixTexFootprint2D (unsigned long long tex, unsigned int texInfo, float x, float y, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DGrad (unsigned long long tex, unsigned int texInfo, float x, float y, float dPdx_x, float dPdx_y, float dPdy_x, float dPdy_y, bool coarse, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DLod (unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool coarse, unsigned int *singleMipLevel)

8.1.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Device side implementation

8.1.2 Macro Definition Documentation

8.1.2.1 OPTIX_DEFINE_optixGetAttribute_BODY

```
#define OPTIX_DEFINE_optixGetAttribute_BODY(
            which )
```

Value:

```
    unsigned int ret;
\
    asm("call (%0), _optix_get_attribute_" #which ", ();" : "=r"(ret) :);
\
    return ret;
```

8.1.2.2 OPTIX_DEFINE_optixGetExceptionDetail_BODY

```
#define OPTIX_DEFINE_optixGetExceptionDetail_BODY(
            which )
```

Value:

```
    unsigned int ret;
\
    asm("call (%0), _optix_get_exception_detail_" #which ", ();" : "=r"(ret) :);
\
    return ret;
```

8.1.3 Function Documentation

8.1.3.1 optixContinuationCall()

```
template<typename ReturnT , typename... ArgTypes>
static __forceinline__ __device__ ReturnT optixContinuationCall (
          unsigned int sbtIndex,
          ArgTypes... args ) [static]
```

8.1.3.2 optixDirectCall()

```
template<typename ReturnT , typename... ArgTypes>
```

```
static __forceinline__ __device__ ReturnT optixDirectCall (
          unsigned int sbtIndex,
          ArgTypes... args ) [static]
```

8.1.3.3 optixGetAttribute_0()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_0 ( ) [static]
```

8.1.3.4 optixGetAttribute_1()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_1 ( ) [static]
```

8.1.3.5 optixGetAttribute_2()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_2 ( ) [static]
```

8.1.3.6 optixGetAttribute_3()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_3 ( ) [static]
```

8.1.3.7 optixGetAttribute_4()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_4 ( ) [static]
```

8.1.3.8 optixGetAttribute_5()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_5 ( ) [static]
```

8.1.3.9 optixGetAttribute_6()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_6 ( ) [static]
```

8.1.3.10 optixGetAttribute_7()

```
static __forceinline__ __device__ unsigned int optixGetAttribute_7 ( ) [static]
```

8.1.3.11 optixGetCatmullRomRocapsVertexData()

```
static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexData (
          float4 data[4] ) [static]
```

8.1.3.12 optixGetCatmullRomRocapsVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetCatmullRomRocapsVertexDataFromHandle (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[4] ) [static]
```

8.1.3.13 optixGetCatmullRomVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetCatmullRomVertexData (
```

```
          float4 data[4] ) [static]
```

8.1.3.14 optixGetCatmullRomVertexData() [2/2]

```
static __forceinline__ __device__ void optixGetCatmullRomVertexData (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[4] ) [static]
```

8.1.3.15 optixGetCatmullRomVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetCatmullRomVertexDataFromHandle (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[4] ) [static]
```

8.1.3.16 optixGetClusterId()

```
static __forceinline__ __device__ unsigned int optixGetClusterId ( ) [static]
```

8.1.3.17 optixGetCubicBezierRocapsVertexData()

```
static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexData (
          float4 data[4] ) [static]
```

8.1.3.18 optixGetCubicBezierRocapsVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetCubicBezierRocapsVertexDataFromHandle (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[4] ) [static]
```

8.1.3.19 optixGetCubicBezierVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetCubicBezierVertexData (
          float4 data[4] ) [static]
```

8.1.3.20 optixGetCubicBezierVertexData() [2/2]

```
static __forceinline__ __device__ void optixGetCubicBezierVertexData (
          OptixTraversableHandle gas,
```

```
            unsigned int primIdx,
            unsigned int sbtGASIndex,
            float time,
            float4 data[4] )  [static]
```

8.1.3.21 optixGetCubicBezierVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetCubicBezierVertexDataFromHandle (
            OptixTraversableHandle gas,
            unsigned int primIdx,
            unsigned int sbtGASIndex,
            float time,
            float4 data[4] )  [static]
```

8.1.3.22 optixGetCubicBSplineRocapsVertexData()

```
static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexData
(
            float4 data[4] )  [static]
```

8.1.3.23 optixGetCubicBSplineRocapsVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetCubicBSplineRocapsVertexDataFromHandle (
            OptixTraversableHandle gas,
            unsigned int primIdx,
            unsigned int sbtGASIndex,
            float time,
            float4 data[4] )  [static]
```

8.1.3.24 optixGetCubicBSplineVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetCubicBSplineVertexData (
            float4 data[4] )  [static]
```

8.1.3.25 optixGetCubicBSplineVertexData() [2/2]

```
static __forceinline__ __device__ void optixGetCubicBSplineVertexData (
            OptixTraversableHandle gas,
            unsigned int primIdx,
            unsigned int sbtGASIndex,
            float time,
            float4 data[4] )  [static]
```

8.1.3.26 optixGetCubicBSplineVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetCubicBSplineVertexDataFromHandle (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[4] )  [static]
```

8.1.3.27 optixGetCurveParameter()

```
static __forceinline__ __device__ float optixGetCurveParameter ( )  [static]
```

8.1.3.28 optixGetExceptionCode()

```
static __forceinline__ __device__ int optixGetExceptionCode ( )  [static]
```

8.1.3.29 optixGetExceptionDetail_0()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0 ( )
[static]
```

8.1.3.30 optixGetExceptionDetail_1()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1 ( )
[static]
```

8.1.3.31 optixGetExceptionDetail_2()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2 ( )
[static]
```

8.1.3.32 optixGetExceptionDetail_3()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3 ( )
[static]
```

8.1.3.33 optixGetExceptionDetail_4()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4 ( )
[static]
```

8.1.3.34 optixGetExceptionDetail_5()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5 ( )
[static]
```

8.1.3.35 optixGetExceptionDetail_6()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6 ( )
[static]
```

8.1.3.36 optixGetExceptionDetail_7()

```
static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7 ( )
```
[static]

8.1.3.37 optixGetExceptionInvalidRay()

```
static __forceinline__ __device__ OptixInvalidRayExceptionDetails
optixGetExceptionInvalidRay ( )
```
[static]

8.1.3.38 optixGetExceptionInvalidSbtOffset()

```
static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset ( )
```
[static]

8.1.3.39 optixGetExceptionInvalidTraversable()

```
static __forceinline__ __device__ OptixTraversableHandle
optixGetExceptionInvalidTraversable ( )
```
[static]

8.1.3.40 optixGetExceptionLineInfo()

```
static __forceinline__ __device__ char * optixGetExceptionLineInfo ( )
```
[static]

8.1.3.41 optixGetExceptionParameterMismatch()

```
static __forceinline__ __device__ OptixParameterMismatchExceptionDetails
optixGetExceptionParameterMismatch ( )
```
[static]

8.1.3.42 optixGetGASMotionStepCount()

```
static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (
          OptixTraversableHandle handle )
```
[static]

8.1.3.43 optixGetGASMotionTimeBegin()

```
static __forceinline__ __device__ float optixGetGASMotionTimeBegin (
          OptixTraversableHandle handle )
```
[static]

8.1.3.44 optixGetGASMotionTimeEnd()

```
static __forceinline__ __device__ float optixGetGASMotionTimeEnd (
          OptixTraversableHandle handle )
```
[static]

8.1.3.45 optixGetGASPointerFromHandle()

```
static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle (
          OptixTraversableHandle handle )
```
[static]

8.1.3.46 optixGetGASTraversableHandle()

```
static __forceinline__ __device__ OptixTraversableHandle
optixGetGASTraversableHandle ( )
```
[static]

8.1.3.47 optixGetHitKind()

```
static __forceinline__ __device__ unsigned int optixGetHitKind ( ) [static]
```

8.1.3.48 optixGetInstanceChildFromHandle()

```
static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle (
          OptixTraversableHandle handle ) [static]
```

8.1.3.49 optixGetInstanceId()

```
static __forceinline__ __device__ unsigned int optixGetInstanceId ( ) [static]
```

8.1.3.50 optixGetInstanceIdFromHandle()

```
static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle
(
          OptixTraversableHandle handle ) [static]
```

8.1.3.51 optixGetInstanceIndex()

```
static __forceinline__ __device__ unsigned int optixGetInstanceIndex ( )
[static]
```

8.1.3.52 optixGetInstanceInverseTransformFromHandle()

```
static __forceinline__ __device__ const float4 *
optixGetInstanceInverseTransformFromHandle (
          OptixTraversableHandle handle ) [static]
```

8.1.3.53 optixGetInstanceTransformFromHandle()

```
static __forceinline__ __device__ const float4 *
optixGetInstanceTransformFromHandle (
          OptixTraversableHandle handle ) [static]
```

8.1.3.54 optixGetInstanceTraversableFromIAS()

```
static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS (
          OptixTraversableHandle ias,
          unsigned int instIdx ) [static]
```

8.1.3.55 optixGetLaunchDimensions()

```
static __forceinline__ __device__ uint3 optixGetLaunchDimensions ( ) [static]
```

8.1.3.56 optixGetLaunchIndex()

```
static __forceinline__ __device__ uint3 optixGetLaunchIndex ( ) [static]
```

8.1.3.57 optixGetLinearCurveVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetLinearCurveVertexData (
          float4 data[2] ) [static]
```

8.1.3.58 optixGetLinearCurveVertexData() [2/2]

```
static __forceinline__ __device__ void optixGetLinearCurveVertexData (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[2] ) [static]
```

8.1.3.59 optixGetLinearCurveVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetLinearCurveVertexDataFromHandle (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[2] ) [static]
```

8.1.3.60 optixGetMatrixMotionTransformFromHandle()

```
static __forceinline__ __device__ const OptixMatrixMotionTransform *
optixGetMatrixMotionTransformFromHandle (
          OptixTraversableHandle handle ) [static]
```

8.1.3.61 optixGetObjectRayDirection()

```
static __forceinline__ __device__ float3 optixGetObjectRayDirection ( )
[static]
```

8.1.3.62 optixGetObjectRayOrigin()

```
static __forceinline__ __device__ float3 optixGetObjectRayOrigin ( ) [static]
```

8.1.3.63 optixGetObjectToWorldTransformMatrix() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix
(
          const HitState & hs,
          float m[12] ) [static]
```

8.1.3.64 optixGetObjectToWorldTransformMatrix() [2/2]

```
static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix
(
          float m[12] ) [static]
```

8.1.3.65 optixGetPayload_0()

```
static __forceinline__ __device__ unsigned int optixGetPayload_0 ( ) [static]
```

8.1.3.66 optixGetPayload_1()

```
static __forceinline__ __device__ unsigned int optixGetPayload_1 ( ) [static]
```

8.1.3.67 optixGetPayload_10()

```
static __forceinline__ __device__ unsigned int optixGetPayload_10 ( ) [static]
```

8.1.3.68 optixGetPayload_11()

```
static __forceinline__ __device__ unsigned int optixGetPayload_11 ( ) [static]
```

8.1.3.69 optixGetPayload_12()

```
static __forceinline__ __device__ unsigned int optixGetPayload_12 ( ) [static]
```

8.1.3.70 optixGetPayload_13()

```
static __forceinline__ __device__ unsigned int optixGetPayload_13 ( ) [static]
```

8.1.3.71 optixGetPayload_14()

```
static __forceinline__ __device__ unsigned int optixGetPayload_14 ( ) [static]
```

8.1.3.72 optixGetPayload_15()

```
static __forceinline__ __device__ unsigned int optixGetPayload_15 ( ) [static]
```

8.1.3.73 optixGetPayload_16()

```
static __forceinline__ __device__ unsigned int optixGetPayload_16 ( ) [static]
```

8.1.3.74 optixGetPayload_17()

```
static __forceinline__ __device__ unsigned int optixGetPayload_17 ( ) [static]
```

8.1.3.75 optixGetPayload_18()

```
static __forceinline__ __device__ unsigned int optixGetPayload_18 ( ) [static]
```

8.1.3.76 optixGetPayload_19()

```
static __forceinline__ __device__ unsigned int optixGetPayload_19 ( ) [static]
```

8.1.3.77 optixGetPayload_2()

```
static __forceinline__ __device__ unsigned int optixGetPayload_2 ( ) [static]
```

8.1.3.78 optixGetPayload_20()

```
static __forceinline__ __device__ unsigned int optixGetPayload_20 ( ) [static]
```

8.1.3.79 optixGetPayload_21()

```
static __forceinline__ __device__ unsigned int optixGetPayload_21 ( ) [static]
```

8.1.3.80 optixGetPayload_22()

```
static __forceinline__ __device__ unsigned int optixGetPayload_22 ( ) [static]
```

8.1.3.81 optixGetPayload_23()

```
static __forceinline__ __device__ unsigned int optixGetPayload_23 ( ) [static]
```

8.1.3.82 optixGetPayload_24()

```
static __forceinline__ __device__ unsigned int optixGetPayload_24 ( ) [static]
```

8.1.3.83 optixGetPayload_25()

```
static __forceinline__ __device__ unsigned int optixGetPayload_25 ( ) [static]
```

8.1.3.84 optixGetPayload_26()

```
static __forceinline__ __device__ unsigned int optixGetPayload_26 ( ) [static]
```

8.1.3.85 optixGetPayload_27()

```
static __forceinline__ __device__ unsigned int optixGetPayload_27 ( ) [static]
```

8.1.3.86 optixGetPayload_28()

```
static __forceinline__ __device__ unsigned int optixGetPayload_28 ( ) [static]
```

8.1.3.87 optixGetPayload_29()

```
static __forceinline__ __device__ unsigned int optixGetPayload_29 ( ) [static]
```

8.1.3.88 optixGetPayload_3()

```
static __forceinline__ __device__ unsigned int optixGetPayload_3 ( ) [static]
```

8.1.3.89 optixGetPayload_30()

```
static __forceinline__ __device__ unsigned int optixGetPayload_30 ( ) [static]
```

8.1.3.90 optixGetPayload_31()

```
static __forceinline__ __device__ unsigned int optixGetPayload_31 ( ) [static]
```

8.1.3.91 optixGetPayload_4()

```
static __forceinline__ __device__ unsigned int optixGetPayload_4 ( ) [static]
```

8.1.3.92 optixGetPayload_5()

```
static __forceinline__ __device__ unsigned int optixGetPayload_5 ( ) [static]
```

8.1.3.93 optixGetPayload_6()

```
static __forceinline__ __device__ unsigned int optixGetPayload_6 ( ) [static]
```

8.1.3.94 optixGetPayload_7()

```
static __forceinline__ __device__ unsigned int optixGetPayload_7 ( ) [static]
```

8.1.3.95 optixGetPayload_8()

```
static __forceinline__ __device__ unsigned int optixGetPayload_8 ( ) [static]
```

8.1.3.96 optixGetPayload_9()

```
static __forceinline__ __device__ unsigned int optixGetPayload_9 ( ) [static]
```

8.1.3.97 optixGetPrimitiveIndex()

```
static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ( )
[static]
```

8.1.3.98 optixGetPrimitiveType() [1/2]

```
static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (
) [static]
```

8.1.3.99 optixGetPrimitiveType() [2/2]

```
static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (
          unsigned int hitKind ) [static]
```

8.1.3.100 optixGetQuadraticBSplineRocapsVertexData()

```
static __forceinline__ __device__ void
optixGetQuadraticBSplineRocapsVertexData (
          float4 data[3] ) [static]
```

8.1.3.101 optixGetQuadraticBSplineRocapsVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetQuadraticBSplineRocapsVertexDataFromHandle (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[3] ) [static]
```

8.1.3.102 optixGetQuadraticBSplineVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (
          float4 data[3] ) [static]
```

8.1.3.103 optixGetQuadraticBSplineVertexData() [2/2]

```
static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[3] ) [static]
```

8.1.3.104 optixGetQuadraticBSplineVertexDataFromHandle()

```
static __forceinline__ __device__ void
optixGetQuadraticBSplineVertexDataFromHandle (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[3] ) [static]
```

8.1.3.105 optixGetRayFlags()

```
static __forceinline__ __device__ unsigned int optixGetRayFlags ( ) [static]
```

8.1.3.106 optixGetRayTime()

```
static __forceinline__ __device__ float optixGetRayTime ( ) [static]
```

8.1.3.107 optixGetRayTmax()

```
static __forceinline__ __device__ float optixGetRayTmax ( ) [static]
```

8.1.3.108 optixGetRayTmin()

```
static __forceinline__ __device__ float optixGetRayTmin ( ) [static]
```

8.1.3.109 optixGetRayVisibilityMask()

```
static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask ( )
[static]
```

8.1.3.110 optixGetRibbonNormal() [1/2]

```
static __forceinline__ __device__ float3 optixGetRibbonNormal (
          float2 ribbonParameters ) [static]
```

8.1.3.111 optixGetRibbonNormal() [2/2]

```
static __forceinline__ __device__ float3 optixGetRibbonNormal (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float2 ribbonParameters ) [static]
```

8.1.3.112 optixGetRibbonNormalFromHandle()

```
static __forceinline__ __device__ float3 optixGetRibbonNormalFromHandle (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float2 ribbonParameters ) [static]
```

8.1.3.113 optixGetRibbonParameters()

```
static __forceinline__ __device__ float2 optixGetRibbonParameters ( ) [static]
```

8.1.3.114 optixGetRibbonVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetRibbonVertexData (
          float4 data[3] ) [static]
```

8.1.3.115 optixGetRibbonVertexData() [2/2]

```
static __forceinline__ __device__ void optixGetRibbonVertexData (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[3] ) [static]
```

8.1.3.116 optixGetRibbonVertexDataFromHandle()

```
static __forceinline__ __device__ void optixGetRibbonVertexDataFromHandle (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[3] ) [static]
```

8.1.3.117 optixGetSbtDataPointer()

```
static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ( )
[static]
```

8.1.3.118 optixGetSbtGASIndex()

```
static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ( ) [static]
```

8.1.3.119 optixGetSphereData() [1/2]

```
static __forceinline__ __device__ void optixGetSphereData (
          float4 data[1] ) [static]
```

8.1.3.120 optixGetSphereData() [2/2]

```
static __forceinline__ __device__ void optixGetSphereData (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[1] ) [static]
```

8.1.3.121 optixGetSphereDataFromHandle()

```
static __forceinline__ __device__ void optixGetSphereDataFromHandle (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float4 data[1] ) [static]
```

8.1.3.122 optixGetSRTMotionTransformFromHandle()

```
static __forceinline__ __device__ const OptixSRTMotionTransform *
optixGetSRTMotionTransformFromHandle (
          OptixTraversableHandle handle ) [static]
```

8.1.3.123 optixGetStaticTransformFromHandle()

```
static __forceinline__ __device__ const OptixStaticTransform *
optixGetStaticTransformFromHandle (
          OptixTraversableHandle handle ) [static]
```

8.1.3.124 optixGetTransformListHandle()

```
static __forceinline__ __device__ OptixTraversableHandle
optixGetTransformListHandle (
          unsigned int index ) [static]
```

8.1.3.125 optixGetTransformListSize()

```
static __forceinline__ __device__ unsigned int optixGetTransformListSize ( )
[static]
```

8.1.3.126 optixGetTransformTypeFromHandle()

```
static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle (
          OptixTraversableHandle handle ) [static]
```

8.1.3.127 optixGetTriangleBarycentrics()

```
static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ( )
[static]
```

8.1.3.128 optixGetTriangleVertexData() [1/2]

```
static __forceinline__ __device__ void optixGetTriangleVertexData (
          float3 data[3] ) [static]
```

8.1.3.129 optixGetTriangleVertexData() [2/2]

```
static __forceinline__ __device__ void optixGetTriangleVertexData (
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float3 data[3] ) [static]
```

8.1.3.130 optixGetTriangleVertexDataFromHandle()

```
static __forceinline__ __device__ void optixGetTriangleVertexDataFromHandle
(
          OptixTraversableHandle gas,
          unsigned int primIdx,
          unsigned int sbtGASIndex,
          float time,
          float3 data[3] ) [static]
```

8.1.3.131 optixGetWorldRayDirection()

```
static __forceinline__ __device__ float3 optixGetWorldRayDirection ( ) [static]
```

8.1.3.132 optixGetWorldRayOrigin()

```
static __forceinline__ __device__ float3 optixGetWorldRayOrigin ( ) [static]
```

8.1.3.133 optixGetWorldToObjectTransformMatrix() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix
(
          const HitState & hs,
          float m[12] ) [static]
```

8.1.3.134 optixGetWorldToObjectTransformMatrix() [2/2]

```
static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix
(
          float m[12] ) [static]
```

8.1.3.135 optixHitObjectGetAttribute_0()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0
( ) [static]
```

8.1.3.136 optixHitObjectGetAttribute_1()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1
( ) [static]
```

8.1.3.137 optixHitObjectGetAttribute_2()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2
( ) [static]
```

8.1.3.138 optixHitObjectGetAttribute_3()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3
( ) [static]
```

8.1.3.139 optixHitObjectGetAttribute_4()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4
( ) [static]
```

8.1.3.140 optixHitObjectGetAttribute_5()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5
( ) [static]
```

8.1.3.141 optixHitObjectGetAttribute_6()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6
( ) [static]
```

8.1.3.142 optixHitObjectGetAttribute_7()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7
( ) [static]
```

8.1.3.143 optixHitObjectGetCatmullRomRocapsVertexData()

```
static __forceinline__ __device__ void
optixHitObjectGetCatmullRomRocapsVertexData (
          float4 data[4] ) [static]
```

8.1.3.144 optixHitObjectGetCatmullRomVertexData()

```
static __forceinline__ __device__ void optixHitObjectGetCatmullRomVertexData
(
          float4 data[4] ) [static]
```

8.1.3.145 optixHitObjectGetClusterId()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetClusterId (
) [static]
```

8.1.3.146 optixHitObjectGetCubicBezierRocapsVertexData()

```
static __forceinline__ __device__ void
optixHitObjectGetCubicBezierRocapsVertexData (
          float4 data[4] ) [static]
```

8.1.3.147 optixHitObjectGetCubicBezierVertexData()

```
static __forceinline__ __device__ void
optixHitObjectGetCubicBezierVertexData (
          float4 data[4] ) [static]
```

8.1.3.148 optixHitObjectGetCubicBSplineRocapsVertexData()

```
static __forceinline__ __device__ void
optixHitObjectGetCubicBSplineRocapsVertexData (
          float4 data[4] ) [static]
```

8.1.3.149 optixHitObjectGetCubicBSplineVertexData()

```
static __forceinline__ __device__ void
optixHitObjectGetCubicBSplineVertexData (
          float4 data[4] ) [static]
```

8.1.3.150 optixHitObjectGetCurveParameter()

```
static __forceinline__ __device__ float optixHitObjectGetCurveParameter ( )
[static]
```

8.1.3.151 optixHitObjectGetGASTraversableHandle()

```
static __forceinline__ __device__ OptixTraversableHandle
optixHitObjectGetGASTraversableHandle ( ) [static]
```

8.1.3.152 optixHitObjectGetHitKind()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind ( )
[static]
```

8.1.3.153 optixHitObjectGetInstanceId()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId (
```

) *[static]*

8.1.3.154 optixHitObjectGetInstanceIndex()

```
static __forceinline__ __device__ unsigned int
optixHitObjectGetInstanceIndex ( ) [static]
```

8.1.3.155 optixHitObjectGetLinearCurveVertexData()

```
static __forceinline__ __device__ void
optixHitObjectGetLinearCurveVertexData (
          float4 data[2] ) [static]
```

8.1.3.156 optixHitObjectGetObjectToWorldTransformMatrix()

```
static __forceinline__ __device__ void
optixHitObjectGetObjectToWorldTransformMatrix (
          float m[12] ) [static]
```

8.1.3.157 optixHitObjectGetPrimitiveIndex()

```
static __forceinline__ __device__ unsigned int
optixHitObjectGetPrimitiveIndex ( ) [static]
```

8.1.3.158 optixHitObjectGetQuadraticBSplineRocapsVertexData()

```
static __forceinline__ __device__ void
optixHitObjectGetQuadraticBSplineRocapsVertexData (
          float4 data[3] ) [static]
```

8.1.3.159 optixHitObjectGetQuadraticBSplineVertexData()

```
static __forceinline__ __device__ void
optixHitObjectGetQuadraticBSplineVertexData (
          float4 data[3] ) [static]
```

8.1.3.160 optixHitObjectGetRayFlags()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetRayFlags ( )
[static]
```

8.1.3.161 optixHitObjectGetRayTime()

```
static __forceinline__ __device__ float optixHitObjectGetRayTime ( ) [static]
```

8.1.3.162 optixHitObjectGetRayTmax()

```
static __forceinline__ __device__ float optixHitObjectGetRayTmax ( ) [static]
```

8.1.3.163 optixHitObjectGetRayTmin()

```
static __forceinline__ __device__ float optixHitObjectGetRayTmin ( ) [static]
```

8.1.3.164 optixHitObjectGetRibbonNormal()

```
static __forceinline__ __device__ float3 optixHitObjectGetRibbonNormal (
          float2 ribbonParameters ) [static]
```

8.1.3.165 optixHitObjectGetRibbonParameters()

```
static __forceinline__ __device__ float2 optixHitObjectGetRibbonParameters (
) [static]
```

8.1.3.166 optixHitObjectGetRibbonVertexData()

```
static __forceinline__ __device__ void optixHitObjectGetRibbonVertexData (
          float4 data[3] ) [static]
```

8.1.3.167 optixHitObjectGetSbtDataPointer()

```
static __forceinline__ __device__ CUdeviceptr
optixHitObjectGetSbtDataPointer ( ) [static]
```

8.1.3.168 optixHitObjectGetSbtGASIndex()

```
static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex
( ) [static]
```

8.1.3.169 optixHitObjectGetSbtRecordIndex()

```
static __forceinline__ __device__ unsigned int
optixHitObjectGetSbtRecordIndex ( ) [static]
```

8.1.3.170 optixHitObjectGetSphereData()

```
static __forceinline__ __device__ void optixHitObjectGetSphereData (
          float4 data[1] ) [static]
```

8.1.3.171 optixHitObjectGetTransformListHandle()

```
static __forceinline__ __device__ OptixTraversableHandle
optixHitObjectGetTransformListHandle (
          unsigned int index ) [static]
```

8.1.3.172 optixHitObjectGetTransformListSize()

```
static __forceinline__ __device__ unsigned int
optixHitObjectGetTransformListSize ( ) [static]
```

8.1.3.173 optixHitObjectGetTraverseData()

```
static __forceinline__ __device__ void optixHitObjectGetTraverseData (
          OptixTraverseData * data ) [static]
```

8.1.3.174 optixHitObjectGetTriangleBarycentrics()

```
static __forceinline__ __device__ float2
optixHitObjectGetTriangleBarycentrics ( ) [static]
```

8.1.3.175 optixHitObjectGetTriangleVertexData()

```
static __forceinline__ __device__ void optixHitObjectGetTriangleVertexData (
          float3 data[3] ) [static]
```

8.1.3.176 optixHitObjectGetWorldRayDirection()

```
static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection
( ) [static]
```

8.1.3.177 optixHitObjectGetWorldRayOrigin()

```
static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin ( )
[static]
```

8.1.3.178 optixHitObjectGetWorldToObjectTransformMatrix()

```
static __forceinline__ __device__ void
optixHitObjectGetWorldToObjectTransformMatrix (
          float m[12] ) [static]
```

8.1.3.179 optixHitObjectIsHit()

```
static __forceinline__ __device__ bool optixHitObjectIsHit ( ) [static]
```

8.1.3.180 optixHitObjectIsMiss()

```
static __forceinline__ __device__ bool optixHitObjectIsMiss ( ) [static]
```

8.1.3.181 optixHitObjectIsNop()

```
static __forceinline__ __device__ bool optixHitObjectIsNop ( ) [static]
```

8.1.3.182 optixHitObjectSetSbtRecordIndex()

```
static __forceinline__ __device__ void optixHitObjectSetSbtRecordIndex (
          unsigned int sbtRecordIndex ) [static]
```

8.1.3.183 optixHitObjectTransformNormalFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixHitObjectTransformNormalFromObjectToWorldSpace (
          float3 normal ) [static]
```

8.1.3.184 optixHitObjectTransformNormalFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixHitObjectTransformNormalFromWorldToObjectSpace (
          float3 normal ) [static]
```

8.1.3.185 optixHitObjectTransformPointFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixHitObjectTransformPointFromObjectToWorldSpace (
          float3 point ) [static]
```

8.1.3.186 optixHitObjectTransformPointFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixHitObjectTransformPointFromWorldToObjectSpace (
          float3 point ) [static]
```

8.1.3.187 optixHitObjectTransformVectorFromObjectToWorldSpace()

```
static __forceinline__ __device__ float3
optixHitObjectTransformVectorFromObjectToWorldSpace (
          float3 vec ) [static]
```

8.1.3.188 optixHitObjectTransformVectorFromWorldToObjectSpace()

```
static __forceinline__ __device__ float3
optixHitObjectTransformVectorFromWorldToObjectSpace (
          float3 vec ) [static]
```

8.1.3.189 optixIgnoreIntersection()

```
static __forceinline__ __device__ void optixIgnoreIntersection ( ) [static]
```

8.1.3.190 optixInvoke() [1/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixInvoke (
          OptixPayloadTypeID type,
          Payload &... payload ) [static]
```

8.1.3.191 optixInvoke() [2/2]

```
template<typename... Payload>
static __forceinline__ __device__ void optixInvoke (
          Payload &... payload ) [static]
```

8.1.3.192 optixIsBackFaceHit() [1/2]

```
static __forceinline__ __device__ bool optixIsBackFaceHit ( ) [static]
```

8.1.3.193 optixIsBackFaceHit() [2/2]

```
static __forceinline__ __device__ bool optixIsBackFaceHit (
          unsigned int hitKind ) [static]
```

8.1.3.194 optixIsFrontFaceHit() [1/2]

```
static __forceinline__ __device__ bool optixIsFrontFaceHit ( ) [static]
```

8.1.3.195 optixIsFrontFaceHit() [2/2]

```
static __forceinline__ __device__ bool optixIsFrontFaceHit (
          unsigned int hitKind ) [static]
```

8.1.3.196 optixIsTriangleBackFaceHit()

```
static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ( ) [static]
```

8.1.3.197 optixIsTriangleFrontFaceHit()

```
static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ( ) [static]
```

8.1.3.198 optixIsTriangleHit()

```
static __forceinline__ __device__ bool optixIsTriangleHit ( ) [static]
```

8.1.3.199 optixMakeHitObject()

```
static __forceinline__ __device__ void optixMakeHitObject (
          OptixTraversableHandle handle,
          float3 rayOrigin,
          float3 rayDirection,
          float tmin,
          float rayTime,
          unsigned int rayFlags,
          OptixTraverseData traverseData,
          const OptixTraversableHandle * transforms,
          unsigned int numTransforms ) [static]
```

8.1.3.200 optixMakeMissHitObject()

```
static __forceinline__ __device__ void optixMakeMissHitObject (
          unsigned int missSBTIndex,
          float3 rayOrigin,
          float3 rayDirection,
          float tmin,
          float tmax,
          float rayTime,
          unsigned int rayFlags ) [static]
```

8.1.3.201 optixMakeNopHitObject()

```
static __forceinline__ __device__ void optixMakeNopHitObject ( ) [static]
```

8.1.3.202 optixReorder() [1/2]

```
static __forceinline__ __device__ void optixReorder ( ) [static]
```

8.1.3.203 optixReorder() [2/2]

```
static __forceinline__ __device__ void optixReorder (
          unsigned int coherenceHint,
          unsigned int numCoherenceHintBits ) [static]
```

8.1.3.204 optixReportIntersection() [1/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
          float hitT,
          unsigned int hitKind ) [static]
```

8.1.3.205 optixReportIntersection() [2/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
          float hitT,
          unsigned int hitKind,
          unsigned int a0 ) [static]
```

8.1.3.206 optixReportIntersection() [3/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
          float hitT,
          unsigned int hitKind,
          unsigned int a0,
          unsigned int a1 ) [static]
```

8.1.3.207 optixReportIntersection() [4/9]

```
static __forceinline__ __device__ bool optixReportIntersection (
          float hitT,
          unsigned int hitKind,
          unsigned int a0,
          unsigned int a1,
          unsigned int a2 ) [static]
```

8.1.3.208 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]
```

8.1.3.209 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]
```

8.1.3.210 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]
```

8.1.3.211 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]
```

8.1.3.212 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]
```

8.1.3.213 optixSetPayload_0()

```
static __forceinline__ __device__ void optixSetPayload_0 (
          unsigned int p )  [static]
```

8.1.3.214 optixSetPayload_1()

```
static __forceinline__ __device__ void optixSetPayload_1 (
          unsigned int p )  [static]
```

8.1.3.215 optixSetPayload_10()

```
static __forceinline__ __device__ void optixSetPayload_10 (
          unsigned int p )  [static]
```

8.1.3.216 optixSetPayload_11()

```
static __forceinline__ __device__ void optixSetPayload_11 (
          unsigned int p )  [static]
```

8.1.3.217 optixSetPayload_12()

```
static __forceinline__ __device__ void optixSetPayload_12 (
          unsigned int p )  [static]
```

8.1.3.218 optixSetPayload_13()

```
static __forceinline__ __device__ void optixSetPayload_13 (
          unsigned int p )  [static]
```

8.1.3.219 optixSetPayload_14()

```
static __forceinline__ __device__ void optixSetPayload_14 (
          unsigned int p )  [static]
```

8.1.3.220 optixSetPayload_15()

```
static __forceinline__ __device__ void optixSetPayload_15 (
          unsigned int p )  [static]
```

8.1.3.221 optixSetPayload_16()

```
static __forceinline__ __device__ void optixSetPayload_16 (
          unsigned int p )  [static]
```

8.1.3.222 optixSetPayload_17()

```
static __forceinline__ __device__ void optixSetPayload_17 (
          unsigned int p )  [static]
```

8.1.3.223 optixSetPayload_18()

```
static __forceinline__ __device__ void optixSetPayload_18 (
          unsigned int p ) [static]
```

8.1.3.224 optixSetPayload_19()

```
static __forceinline__ __device__ void optixSetPayload_19 (
          unsigned int p ) [static]
```

8.1.3.225 optixSetPayload_2()

```
static __forceinline__ __device__ void optixSetPayload_2 (
          unsigned int p ) [static]
```

8.1.3.226 optixSetPayload_20()

```
static __forceinline__ __device__ void optixSetPayload_20 (
          unsigned int p ) [static]
```

8.1.3.227 optixSetPayload_21()

```
static __forceinline__ __device__ void optixSetPayload_21 (
          unsigned int p ) [static]
```

8.1.3.228 optixSetPayload_22()

```
static __forceinline__ __device__ void optixSetPayload_22 (
          unsigned int p ) [static]
```

8.1.3.229 optixSetPayload_23()

```
static __forceinline__ __device__ void optixSetPayload_23 (
          unsigned int p ) [static]
```

8.1.3.230 optixSetPayload_24()

```
static __forceinline__ __device__ void optixSetPayload_24 (
          unsigned int p ) [static]
```

8.1.3.231 optixSetPayload_25()

```
static __forceinline__ __device__ void optixSetPayload_25 (
          unsigned int p ) [static]
```

8.1.3.232 optixSetPayload_26()

```
static __forceinline__ __device__ void optixSetPayload_26 (
          unsigned int p ) [static]
```

8.1.3.233 optixSetPayload_27()

```
static __forceinline__ __device__ void optixSetPayload_27 (
```

```
          unsigned int p ) [static]
```

8.1.3.234 optixSetPayload_28()

```
static __forceinline__ __device__ void optixSetPayload_28 (
          unsigned int p ) [static]
```

8.1.3.235 optixSetPayload_29()

```
static __forceinline__ __device__ void optixSetPayload_29 (
          unsigned int p ) [static]
```

8.1.3.236 optixSetPayload_3()

```
static __forceinline__ __device__ void optixSetPayload_3 (
          unsigned int p ) [static]
```

8.1.3.237 optixSetPayload_30()

```
static __forceinline__ __device__ void optixSetPayload_30 (
          unsigned int p ) [static]
```

8.1.3.238 optixSetPayload_31()

```
static __forceinline__ __device__ void optixSetPayload_31 (
          unsigned int p ) [static]
```

8.1.3.239 optixSetPayload_4()

```
static __forceinline__ __device__ void optixSetPayload_4 (
          unsigned int p ) [static]
```

8.1.3.240 optixSetPayload_5()

```
static __forceinline__ __device__ void optixSetPayload_5 (
          unsigned int p ) [static]
```

8.1.3.241 optixSetPayload_6()

```
static __forceinline__ __device__ void optixSetPayload_6 (
          unsigned int p ) [static]
```

8.1.3.242 optixSetPayload_7()

```
static __forceinline__ __device__ void optixSetPayload_7 (
          unsigned int p ) [static]
```

8.1.3.243 optixSetPayload_8()

```
static __forceinline__ __device__ void optixSetPayload_8 (
          unsigned int p ) [static]
```

8.1.3.244 optixSetPayload_9()

```
static __forceinline__ __device__ void optixSetPayload_9 (
          unsigned int p ) [static]
```

8.1.3.245 optixSetPayloadTypes()

```
static __forceinline__ __device__ void optixSetPayloadTypes (
          unsigned int types ) [static]
```

8.1.3.246 optixTerminateRay()

```
static __forceinline__ __device__ void optixTerminateRay ( ) [static]
```

8.1.3.247 optixTexFootprint2D()

```
static __forceinline__ __device__ uint4 optixTexFootprint2D (
          unsigned long long tex,
          unsigned int texInfo,
          float x,
          float y,
          unsigned int * singleMipLevel ) [static]
```

8.1.3.248 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]
```

8.1.3.249 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]
```

8.1.3.250 optixThrowException() [1/9]

```
static __forceinline__ __device__ void optixThrowException (
          int exceptionCode ) [static]
```

8.1.3.251 optixThrowException() [2/9]

```
static __forceinline__ __device__ void optixThrowException (
          int exceptionCode,
          unsigned int exceptionDetail0 ) [static]
```

8.1.3.252 optixThrowException() [3/9]

```
static __forceinline__ __device__ void optixThrowException (
          int exceptionCode,
          unsigned int exceptionDetail0,
          unsigned int exceptionDetail1 ) [static]
```

8.1.3.253 optixThrowException() [4/9]

```
static __forceinline__ __device__ void optixThrowException (
          int exceptionCode,
          unsigned int exceptionDetail0,
          unsigned int exceptionDetail1,
          unsigned int exceptionDetail2 ) [static]
```

8.1.3.254 optixThrowException() [5/9]

```
static __forceinline__ __device__ void optixThrowException (
          int exceptionCode,
          unsigned int exceptionDetail0,
          unsigned int exceptionDetail1,
          unsigned int exceptionDetail2,
          unsigned int exceptionDetail3 ) [static]
```

8.1.3.255 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]
```

8.1.3.256 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]
```

8.1.3.257 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]
```

8.1.3.258 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]
```

8.1.3.259 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]
```

8.1.3.260 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]
```

8.1.3.261 optixTransformNormalFromObjectToWorldSpace() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformNormalFromObjectToWorldSpace (
            const HitState & hs,
            float3 normal ) [static]
```

8.1.3.262 optixTransformNormalFromObjectToWorldSpace() [2/2]

```
static __forceinline__ __device__ float3
optixTransformNormalFromObjectToWorldSpace (
            float3 normal ) [static]
```

8.1.3.263 optixTransformNormalFromWorldToObjectSpace() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformNormalFromWorldToObjectSpace (
            const HitState & hs,
            float3 normal ) [static]
```

8.1.3.264 optixTransformNormalFromWorldToObjectSpace() [2/2]

```
static __forceinline__ __device__ float3
optixTransformNormalFromWorldToObjectSpace (
          float3 normal ) [static]
```

8.1.3.265 optixTransformPointFromObjectToWorldSpace() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformPointFromObjectToWorldSpace (
          const HitState & hs,
          float3 point ) [static]
```

8.1.3.266 optixTransformPointFromObjectToWorldSpace() [2/2]

```
static __forceinline__ __device__ float3
optixTransformPointFromObjectToWorldSpace (
          float3 point ) [static]
```

8.1.3.267 optixTransformPointFromWorldToObjectSpace() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformPointFromWorldToObjectSpace (
          const HitState & hs,
          float3 point ) [static]
```

8.1.3.268 optixTransformPointFromWorldToObjectSpace() [2/2]

```
static __forceinline__ __device__ float3
optixTransformPointFromWorldToObjectSpace (
          float3 point ) [static]
```

8.1.3.269 optixTransformVectorFromObjectToWorldSpace() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformVectorFromObjectToWorldSpace (
          const HitState & hs,
          float3 vec ) [static]
```

8.1.3.270 optixTransformVectorFromObjectToWorldSpace() [2/2]

```
static __forceinline__ __device__ float3
optixTransformVectorFromObjectToWorldSpace (
          float3 vec ) [static]
```

8.1.3.271 optixTransformVectorFromWorldToObjectSpace() [1/2]

```
template<typename HitState >
static __forceinline__ __device__ float3
optixTransformVectorFromWorldToObjectSpace (
          const HitState & hs,
          float3 vec ) [static]
```

8.1.3.272 optixTransformVectorFromWorldToObjectSpace() [2/2]

```
static __forceinline__ __device__ float3
optixTransformVectorFromWorldToObjectSpace (
          float3 vec ) [static]
```

8.1.3.273 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]
```

8.1.3.274 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]
```

8.1.3.275 optixUndefinedValue()

```
static __forceinline__ __device__ unsigned int optixUndefinedValue ( ) [static]
```

8.2 optix_device_impl.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
4 *
5 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
6 * property and proprietary rights in and to this material, related
7 * documentation and any modifications thereto. Any use, reproduction,
8 * disclosure or distribution of this material and related documentation
9 * without an express license agreement from NVIDIA CORPORATION or
10 * its affiliates is strictly prohibited.
11 */
20 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
21 #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.")
22 #endif
23
24 #ifndef OPTIX_OPTIX_DEVICE_IMPL_H
25 #define OPTIX_OPTIX_DEVICE_IMPL_H
26
27 #include "internal/optix_device_impl_transformations.h"
28
29 #ifndef __CUDACC_RTC__
30 #include <initializer_list>
31 #include <type_traits>
32 #endif
33
34 namespace optix_internal {
35 template <typename...>
36 struct TypePack{};
37 }  // namespace optix_internal
38
39 template <typename... Payload>
40 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
41                                                   float3                rayOrigin,
42                                                   float3                rayDirection,
43                                                   float                 tmin,
44                                                   float                 tmax,
45                                                   float                 rayTime,
46                                                   OptixVisibilityMask   visibilityMask,
47                                                   unsigned int          rayFlags,
48                                                   unsigned int          SBToffset,
49                                                   unsigned int          SBTstride,
50                                                   unsigned int          missSBTIndex,
51                                                   Payload&...             payload)
52 {
53     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
54     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
55     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
56     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
57 #ifndef __CUDACC_RTC__
58     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int»::value,
59                   "All payload parameters need to be unsigned int.");
60 #endif
61
```

```
62     OptixPayloadTypeID type = OPTIX_PAYLOAD_TYPE_DEFAULT;
63     float              ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
64     float              dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
65     // The first entry in the initializer list is necessary to make empty payload work.
66     unsigned int p[33]       = { 0, payload... };
67     int          payloadSize = (int)sizeof...(Payload);
68     asm volatile(
69         "call"
70
"(%0,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18,%19,%20,%21,%22,%23,%24,%25,%26,%27,%28,%"
71         "29,%30,%31),"
72         "_optix_trace_typed_32,"
73
"(%32,%33,%34,%35,%36,%37,%38,%39,%40,%41,%42,%43,%44,%45,%46,%47,%48,%49,%50,%51,%52,%53,%54,%55,%56,%57,%58,
74         "59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
75         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
76           "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
77           "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
78           "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
79           "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
80         : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
81           "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBToffset), "r"(SBTstride),
82           "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
83           "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
84           "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
85           "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
86           "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
87         :);
88     // The initializer list only exists to force pack expansion.
89     // Initializer lists guarantee strict left to right evaluation, so using index++ is safe.
90     // Conceptually this expands to { payload_0 = p[index++], payload_1 = p[index++], ... payload_N =
p[index++] }
91     unsigned int index = 1;
92     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
93 }
94
95 template <typename... Payload>
96 static __forceinline__ __device__ void optixTraverse(OptixTraversableHandle handle,
97                                                      float3                 rayOrigin,
98                                                      float3                 rayDirection,
99                                                      float                  tmin,
100                                                     float                  tmax,
101                                                     float                  rayTime,
102                                                     OptixVisibilityMask    visibilityMask,
103                                                     unsigned int           rayFlags,
104                                                     unsigned int           SBToffset,
105                                                     unsigned int           SBTstride,
106                                                     unsigned int           missSBTIndex,
107                                                     Payload&... payload)
108 {
109     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
110     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
111     // TypePack 1    unsigned int    T0      T1      T2    ...    Tn-1        Tn
112     // TypePack 2      T0            T1      T2      T3    ...    Tn        unsigned int
113 #ifndef __CUDACC_RTC__
114     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int»::value,
115                   "All payload parameters need to be unsigned int.");
116 #endif
117
118     OptixPayloadTypeID type = OPTIX_PAYLOAD_TYPE_DEFAULT;
119     float              ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
120     float              dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
121     // The first entry in the initializer list is necessary to make empty payload work.
122     unsigned int p[33]       = {0, payload...};
123     int          payloadSize = (int)sizeof...(Payload);
124     asm volatile(
```

```
125         "call"
126
"(%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,%"
127         "29,%30,%31),"
128         "_optix_hitobject_traverse,"
129
"(%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,
130         "59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
131         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
132           "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
133           "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
134           "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
135           "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
136         : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
137           "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBToffset), "r"(SBTstride),
138           "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
139           "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
140           "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
141           "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
142           "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
143         :);
144     // The initializer list only exists to force pack expansion.
145     // Initializer lists guarantee strict left to right evaluation, so using index++ is safe.
146     // Conceptually this expands to { payload_0 = p[index++], payload_1 = p[index++], ... payload_N =
p[index++] }
147     unsigned int index = 1;
148     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
149 }
150
151 template <typename... Payload>
152 static __forceinline__ __device__ void optixTrace(OptixPayloadTypeID     type,
153                                                   OptixTraversableHandle handle,
154                                                   float3                 rayOrigin,
155                                                   float3                 rayDirection,
156                                                   float                  tmin,
157                                                   float                  tmax,
158                                                   float                  rayTime,
159                                                   OptixVisibilityMask    visibilityMask,
160                                                   unsigned int           rayFlags,
161                                                   unsigned int           SBToffset,
162                                                   unsigned int           SBTstride,
163                                                   unsigned int           missSBTIndex,
164                                                   Payload&...              payload)
165 {
166     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
167     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
168     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
169     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
170 #ifndef __CUDACC_RTC__
171     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int»::value,
172                   "All payload parameters need to be unsigned int.");
173 #endif
174
175     float        ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
176     float        dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
177     unsigned int p[33]       = {0, payload...};
178     int          payloadSize = (int)sizeof...(Payload);
179
180     asm volatile(
181         "call"
182
"(%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,%"
183         "29,%30,%31),"
184         "_optix_trace_typed_32,"
185
"(%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,
```

```
186         "59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
187         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
188           "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
189           "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
190           "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
191           "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
192         : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
193           "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBToffset), "r"(SBTstride),
194           "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
195           "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
196           "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
197           "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
198           "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
199         :);
200     // The initializer list only exists to force pack expansion.
201     // Initializer lists guarantee strict left to right evaluation, so using index++ is safe.
202     // Conceptually this expands to { payload_0 = p[index++], payload_1 = p[index++], ... payload_N =
p[index++] }
203     unsigned int index = 1;
204     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
205 }
206
207 template <typename... Payload>
208 static __forceinline__ __device__ void optixTraverse(OptixPayloadTypeID     type,
209                                                      OptixTraversableHandle handle,
210                                                      float3                 rayOrigin,
211                                                      float3                 rayDirection,
212                                                      float                  tmin,
213                                                      float                  tmax,
214                                                      float                  rayTime,
215                                                      OptixVisibilityMask    visibilityMask,
216                                                      unsigned int           rayFlags,
217                                                      unsigned int           SBToffset,
218                                                      unsigned int           SBTstride,
219                                                      unsigned int           missSBTIndex,
220                                                      Payload&... payload)
221 {
222     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
223     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
224     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
225     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
226 #ifndef __CUDACC_RTC__
227     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int»::value,
228                   "All payload parameters need to be unsigned int.");
229 #endif
230
231     float        ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
232     float        dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
233     unsigned int p[33]       = {0, payload...};
234     int          payloadSize = (int)sizeof...(Payload);
235     asm volatile(
236         "call"
237
"(%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,%"
238         "29,%30,%31),"
239         "_optix_hitobject_traverse,"
240
"(%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,
241         "59,%60,%61,%62,%63,%64,%65,%66,%67,%68,%69,%70,%71,%72,%73,%74,%75,%76,%77,%78,%79,%80);"
242         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
243           "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
244           "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
245           "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
246           "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
247         : "r"(type), "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
248           "f"(tmax), "f"(rayTime), "r"(visibilityMask), "r"(rayFlags), "r"(SBToffset), "r"(SBTstride),
```

```
249           "r"(missSBTIndex), "r"(payloadSize), "r"(p[1]), "r"(p[2]), "r"(p[3]), "r"(p[4]), "r"(p[5]),
250           "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]), "r"(p[11]), "r"(p[12]), "r"(p[13]),
251           "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]), "r"(p[18]), "r"(p[19]), "r"(p[20]),
252           "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]), "r"(p[25]), "r"(p[26]), "r"(p[27]),
253           "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
254         :);
255     // The initializer list only exists to force pack expansion.
256     // Initializer lists guarantee strict left to right evaluation, so using index++ is safe.
257     // Conceptually this expands to { payload_0 = p[index++], payload_1 = p[index++], ... payload_N =
p[index++] }
258     unsigned int index = 1;
259     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
260 }
261
262 static __forceinline__ __device__ void optixReorder(unsigned int coherenceHint, unsigned int
numCoherenceHintBits)
263 {
264     asm volatile(
265          "call"
266          "(),"
267          "_optix_hitobject_reorder,"
268          "(%0,%1);"
269          :
270          : "r"(coherenceHint), "r"(numCoherenceHintBits)
271          :);
272 }
273
274 static __forceinline__ __device__ void optixReorder()
275 {
276     unsigned int coherenceHint        = 0;
277     unsigned int numCoherenceHintBits = 0;
278     asm volatile(
279          "call"
280          "(),"
281          "_optix_hitobject_reorder,"
282          "(%0,%1);"
283          :
284          : "r"(coherenceHint), "r"(numCoherenceHintBits)
285          :);
286 }
287
288 template <typename... Payload>
289 static __forceinline__ __device__ void optixInvoke(OptixPayloadTypeID type, Payload&... payload)
290 {
291     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
292     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
293     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
294     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
295 #ifndef __CUDACC_RTC__
296     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int»::value,
297                   "All payload parameters need to be unsigned int.");
298 #endif
299
300     unsigned int p[33]       = {0, payload...};
301     int          payloadSize = (int)sizeof...(Payload);
302
303     asm volatile(
304         "call"
305
"(%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,%"
306          "29,%30,%31),"
307          "_optix_hitobject_invoke,"
308
"(%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,
309          "59,%60,%61,%62,%63,%64,%65);"
310          : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
```

```
311           "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
312           "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
313           "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
314           "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
315         : "r"(type), "r"(payloadSize), "r"(p[1]), "r"(p[2]),
316           "r"(p[3]), "r"(p[4]), "r"(p[5]), "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]),
317           "r"(p[11]), "r"(p[12]), "r"(p[13]), "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]),
318           "r"(p[18]), "r"(p[19]), "r"(p[20]), "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]),
319           "r"(p[25]), "r"(p[26]), "r"(p[27]), "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
320         :);
321
322     // The initializer list only exists to force pack expansion.
323     // Initializer lists guarantee strict left to right evaluation, so using index++ is safe.
324     // Conceptually this expands to { payload_0 = p[index++], payload_1 = p[index++], ... payload_N =
p[index++] }
325     unsigned int index = 1;
326     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
327 }
328
329 template <typename... Payload>
330 static __forceinline__ __device__ void optixInvoke(Payload&... payload)
331 {
332     // std::is_same compares each type in the two TypePacks to make sure that all types are unsigned int.
333     // TypePack 1    unsigned int    T0      T1      T2   ...    Tn-1        Tn
334     // TypePack 2      T0            T1      T2      T3   ...    Tn        unsigned int
335     static_assert(sizeof...(Payload) <= 32, "Only up to 32 payload values are allowed.");
336 #ifndef __CUDACC_RTC__
337     static_assert(std::is_same<optix_internal::TypePack<unsigned int, Payload...>,
optix_internal::TypePack<Payload..., unsigned int»::value,
338                   "All payload parameters need to be unsigned int.");
339 #endif
340
341     OptixPayloadTypeID type        = OPTIX_PAYLOAD_TYPE_DEFAULT;
342     unsigned int       p[33]       = {0, payload...};
343     int                payloadSize = (int)sizeof...(Payload);
344
345     asm volatile(
346         "call"
347
"(%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,%"
348         "29,%30,%31),"
349         "_optix_hitobject_invoke,"
350
"(%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,
351         "59,%60,%61,%62,%63,%64,%65);"
352         : "=r"(p[1]), "=r"(p[2]), "=r"(p[3]), "=r"(p[4]), "=r"(p[5]), "=r"(p[6]), "=r"(p[7]),
353           "=r"(p[8]), "=r"(p[9]), "=r"(p[10]), "=r"(p[11]), "=r"(p[12]), "=r"(p[13]), "=r"(p[14]),
354           "=r"(p[15]), "=r"(p[16]), "=r"(p[17]), "=r"(p[18]), "=r"(p[19]), "=r"(p[20]), "=r"(p[21]),
355           "=r"(p[22]), "=r"(p[23]), "=r"(p[24]), "=r"(p[25]), "=r"(p[26]), "=r"(p[27]), "=r"(p[28]),
356           "=r"(p[29]), "=r"(p[30]), "=r"(p[31]), "=r"(p[32])
357         : "r"(type), "r"(payloadSize), "r"(p[1]), "r"(p[2]),
358           "r"(p[3]), "r"(p[4]), "r"(p[5]), "r"(p[6]), "r"(p[7]), "r"(p[8]), "r"(p[9]), "r"(p[10]),
359           "r"(p[11]), "r"(p[12]), "r"(p[13]), "r"(p[14]), "r"(p[15]), "r"(p[16]), "r"(p[17]),
360           "r"(p[18]), "r"(p[19]), "r"(p[20]), "r"(p[21]), "r"(p[22]), "r"(p[23]), "r"(p[24]),
361           "r"(p[25]), "r"(p[26]), "r"(p[27]), "r"(p[28]), "r"(p[29]), "r"(p[30]), "r"(p[31]), "r"(p[32])
362         :);
363
364     // The initializer list only exists to force pack expansion.
365     // Initializer lists guarantee strict left to right evaluation, so using index++ is safe.
366     // Conceptually this expands to { payload_0 = p[index++], payload_1 = p[index++], ... payload_N =
p[index++] }
367     unsigned int index = 1;
368     (void)std::initializer_list<unsigned int>{index, (payload = p[index++])...};
369 }
370
371 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle        handle,
372                                                           float3                         rayOrigin,
```

```
373                                                         float3                        rayDirection,
374                                                         float                         tmin,
375                                                         float                         rayTime,
376                                                         unsigned int                  rayFlags,
377                                                         OptixTraverseData             traverseData,
378                                                         const OptixTraversableHandle* transforms,
379                                                         unsigned int                  numTransforms)
380 {
381     float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
382     float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
383
384     asm volatile(
385         "call"
386         "(),"
387         "_optix_hitobject_make_with_traverse_data_v2,"
388
"(%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,%2
389         :
390         : "l"(handle), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin), "f"(rayTime),
"r"(rayFlags),
391           "r"(traverseData.data[0]), "r"(traverseData.data[1]), "r"(traverseData.data[2]),
392           "r"(traverseData.data[3]), "r"(traverseData.data[4]), "r"(traverseData.data[5]),
393           "r"(traverseData.data[6]), "r"(traverseData.data[7]), "r"(traverseData.data[8]),
394           "r"(traverseData.data[9]), "r"(traverseData.data[10]), "r"(traverseData.data[11]),
395           "r"(traverseData.data[12]), "r"(traverseData.data[13]), "r"(traverseData.data[14]),
396           "r"(traverseData.data[15]), "r"(traverseData.data[16]), "r"(traverseData.data[17]),
397           "r"(traverseData.data[18]), "r"(traverseData.data[19]), "l"(transforms), "r"(numTransforms)
398         );
399 }
400
401  static __forceinline__ __device__ void optixMakeMissHitObject(unsigned int missSBTIndex,
402                                                                float3       rayOrigin,
403                                                                float3       rayDirection,
404                                                                float        tmin,
405                                                                float        tmax,
406                                                                float        rayTime,
407                                                                unsigned int rayFlags)
408 {
409     float ox = rayOrigin.x, oy = rayOrigin.y, oz = rayOrigin.z;
410     float dx = rayDirection.x, dy = rayDirection.y, dz = rayDirection.z;
411
412     asm volatile(
413          "call"
414          "(),"
415          "_optix_hitobject_make_miss_v2,"
416          "(%0,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10);"
417          :
418          : "r"(missSBTIndex), "f"(ox), "f"(oy), "f"(oz), "f"(dx), "f"(dy), "f"(dz), "f"(tmin),
419            "f"(tmax), "f"(rayTime), "r"(rayFlags)
420          );
421 }
422
423 static __forceinline__ __device__ void optixMakeNopHitObject( )
424 {
425     asm volatile(
426          "call"
427          "(),"
428          "_optix_hitobject_make_nop,"
429          "();"
430          :
431          :
432          );
433 }
434
435 static __forceinline__ __device__ void optixHitObjectGetTraverseData(OptixTraverseData* data)
436 {
437     asm volatile(
```

```
438           "call"
439           "(%0,%1,%2,%3,%4,%5,%6,%7,%8,%9,%10,%11,%12,%13,%14,%15,%16,%17,%18,%19),"
440           "_optix_hitobject_get_traverse_data,"
441           "();"
442           : "=r"(data->data[0]), "=r"(data->data[1]), "=r"(data->data[2]), "=r"(data->data[3]),
"=r"(data->data[4]),
443             "=r"(data->data[5]), "=r"(data->data[6]), "=r"(data->data[7]), "=r"(data->data[8]),
"=r"(data->data[9]),
444             "=r"(data->data[10]), "=r"(data->data[11]), "=r"(data->data[12]), "=r"(data->data[13]),
"=r"(data->data[14]),
445             "=r"(data->data[15]), "=r"(data->data[16]), "=r"(data->data[17]), "=r"(data->data[18]),
"=r"(data->data[19])
446           :
447           :);
448 }
449
450 static __forceinline__ __device__ bool optixHitObjectIsHit()
451 {
452     unsigned int result;
453     asm volatile(
454           "call (%0), _optix_hitobject_is_hit,"
455           "();"
456           : "=r"(result)
457           :
458           :);
459     return result;
460 }
461
462 static __forceinline__ __device__ bool optixHitObjectIsMiss()
463 {
464     unsigned int result;
465     asm volatile(
466           "call (%0), _optix_hitobject_is_miss,"
467           "();"
468           : "=r"(result)
469           :
470           :);
471     return result;
472 }
473
474 static __forceinline__ __device__ bool optixHitObjectIsNop()
475 {
476     unsigned int result;
477     asm volatile(
478           "call (%0), _optix_hitobject_is_nop,"
479           "();"
480           : "=r"(result)
481           :
482           :);
483     return result;
484 }
485
486 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId()
487 {
488     unsigned int result;
489     asm volatile(
490           "call (%0), _optix_hitobject_get_instance_id,"
491           "();"
492           : "=r"(result)
493           :
494           :);
495     return result;
496 }
497
498 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex()
499 {
500     unsigned int result;
```

```
501     asm volatile(
502          "call (%0), _optix_hitobject_get_instance_idx,"
503          "();"
504          : "=r"(result)
505          :
506          :);
507     return result;
508 }
509
510 static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex()
511 {
512     unsigned int result;
513     asm volatile(
514          "call (%0), _optix_hitobject_get_primitive_idx,"
515          "();"
516          : "=r"(result)
517          :
518          :);
519     return result;
520 }
521
522 static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize()
523 {
524     unsigned int result;
525     asm volatile(
526          "call (%0), _optix_hitobject_get_transform_list_size,"
527          "();"
528          : "=r"(result)
529          :
530          :);
531     return result;
532 }
533
534 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle(unsigned
int index)
535 {
536     unsigned long long result;
537     asm volatile(
538          "call (%0), _optix_hitobject_get_transform_list_handle,"
539          "(%1);"
540          : "=l"(result)
541          : "r"(index)
542          :);
543     return result;
544 }
545
546 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex()
547 {
548     unsigned int result;
549     asm volatile(
550          "call (%0), _optix_hitobject_get_sbt_gas_idx,"
551          "();"
552          : "=r"(result)
553          :
554          :);
555     return result;
556 }
557
558 static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind()
559 {
560     unsigned int result;
561     asm volatile(
562          "call (%0), _optix_hitobject_get_hitkind,"
563          "();"
564          : "=r"(result)
565          :
566          :);
```

```
567     return result;
568 }
569
570 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin( )
571 {
572     float x, y, z;
573     asm volatile(
574          "call (%0), _optix_hitobject_get_world_ray_origin_x,"
575          "();"
576          : "=f"(x)
577          :
578          :);
579     asm volatile(
580          "call (%0), _optix_hitobject_get_world_ray_origin_y,"
581          "();"
582          : "=f"(y)
583          :
584          :);
585     asm volatile(
586          "call (%0), _optix_hitobject_get_world_ray_origin_z,"
587          "();"
588          : "=f"(z)
589          :
590          :);
591     return make_float3(x, y, z);
592 }
593
594 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection( )
595 {
596     float x, y, z;
597     asm volatile(
598          "call (%0), _optix_hitobject_get_world_ray_direction_x,"
599          "();"
600          : "=f"(x)
601          :
602          :);
603     asm volatile(
604          "call (%0), _optix_hitobject_get_world_ray_direction_y,"
605          "();"
606          : "=f"(y)
607          :
608          :);
609     asm volatile(
610          "call (%0), _optix_hitobject_get_world_ray_direction_z,"
611          "();"
612          : "=f"(z)
613          :
614          :);
615     return make_float3(x, y, z);
616 }
617
618 static __forceinline__ __device__ float optixHitObjectGetRayTmin( )
619 {
620     float result;
621     asm volatile(
622          "call (%0), _optix_hitobject_get_ray_tmin,"
623          "();"
624          : "=f"(result)
625          :
626          :);
627     return result;
628 }
629
630 static __forceinline__ __device__ float optixHitObjectGetRayTmax( )
631 {
632     float result;
633     asm volatile(
```

```
634          "call (%0), _optix_hitobject_get_ray_tmax,"
635          "();"
636          : "=f"(result)
637          :
638          :);
639      return result;
640  }
641
642  static __forceinline__ __device__ float optixHitObjectGetRayTime()
643  {
644      float result;
645      asm volatile(
646          "call (%0), _optix_hitobject_get_ray_time,"
647          "();"
648          : "=f"(result)
649          :
650          :);
651      return result;
652  }
653
654  static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0()
655  {
656      unsigned int ret;
657      asm volatile(
658          "call (%0), _optix_hitobject_get_attribute,"
659          "(%1);"
660          : "=r"(ret)
661          : "r"(0)
662          :);
663      return ret;
664  }
665
666  static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1()
667  {
668      unsigned int ret;
669      asm volatile(
670          "call (%0), _optix_hitobject_get_attribute,"
671          "(%1);"
672          : "=r"(ret)
673          : "r"(1)
674          :);
675      return ret;
676  }
677
678  static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2()
679  {
680      unsigned int ret;
681      asm volatile(
682          "call (%0), _optix_hitobject_get_attribute,"
683          "(%1);"
684          : "=r"(ret)
685          : "r"(2)
686          :);
687      return ret;
688  }
689
690  static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3()
691  {
692      unsigned int ret;
693      asm volatile(
694          "call (%0), _optix_hitobject_get_attribute,"
695          "(%1);"
696          : "=r"(ret)
697          : "r"(3)
698          :);
699      return ret;
700  }
```

```
701
702 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4()
703 {
704     unsigned int ret;
705     asm volatile(
706          "call (%0), _optix_hitobject_get_attribute,"
707          "(%1);"
708          : "=r"(ret)
709          : "r"(4)
710          :);
711     return ret;
712 }
713
714 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5()
715 {
716     unsigned int ret;
717     asm volatile(
718          "call (%0), _optix_hitobject_get_attribute,"
719          "(%1);"
720          : "=r"(ret)
721          : "r"(5)
722          :);
723     return ret;
724 }
725
726 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6()
727 {
728     unsigned int ret;
729     asm volatile(
730          "call (%0), _optix_hitobject_get_attribute,"
731          "(%1);"
732          : "=r"(ret)
733          : "r"(6)
734          :);
735     return ret;
736 }
737
738 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7()
739 {
740     unsigned int ret;
741     asm volatile(
742          "call (%0), _optix_hitobject_get_attribute,"
743          "(%1);"
744          : "=r"(ret)
745          : "r"(7)
746          :);
747     return ret;
748 }
749
750 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex()
751 {
752     unsigned int result;
753     asm volatile(
754          "call (%0), _optix_hitobject_get_sbt_record_index,"
755          "();"
756          : "=r"(result)
757          :
758          :);
759     return result;
760 }
761
762 static __forceinline__ __device__ void optixHitObjectSetSbtRecordIndex(unsigned int sbtRecordIndex)
763 {
764     asm volatile(
765         "call (), _optix_hitobject_set_sbt_record_index,"
766         "(%0);"
767         :
```

```
768         : "r"(sbtRecordIndex)
769         :);
770 }
771
772 static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer()
773 {
774     unsigned long long ptr;
775     asm volatile(
776          "call (%0), _optix_hitobject_get_sbt_data_pointer,"
777          "();"
778          : "=l"(ptr)
779          :
780          :);
781     return ptr;
782 }
783
784
785 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetGASTraversableHandle()
786 {
787     unsigned long long handle;
788     asm("call (%0), _optix_hitobject_get_gas_traversable_handle, ();" : "=l"(handle) :);
789     return (OptixTraversableHandle)handle;
790 }
791
792
793 static __forceinline__ __device__ unsigned int optixHitObjectGetRayFlags()
794 {
795     unsigned int u0;
796     asm("call (%0), _optix_hitobject_get_ray_flags, ();" : "=r"(u0) :);
797     return u0;
798 }
799
800
801 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p)
802 {
803     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(0), "r"(p) :);
804 }
805
806 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p)
807 {
808     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(1), "r"(p) :);
809 }
810
811 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p)
812 {
813     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(2), "r"(p) :);
814 }
815
816 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p)
817 {
818     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(3), "r"(p) :);
819 }
820
821 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p)
822 {
823     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(4), "r"(p) :);
824 }
825
826 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p)
827 {
828     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(5), "r"(p) :);
829 }
830
831 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p)
832 {
833     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(6), "r"(p) :);
834 }
```

```
835
836 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p)
837 {
838     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(7), "r"(p) :);
839 }
840
841 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p)
842 {
843     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(8), "r"(p) :);
844 }
845
846 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p)
847 {
848     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(9), "r"(p) :);
849 }
850
851 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p)
852 {
853     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(10), "r"(p) :);
854 }
855
856 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p)
857 {
858     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(11), "r"(p) :);
859 }
860
861 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p)
862 {
863     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(12), "r"(p) :);
864 }
865
866 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p)
867 {
868     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(13), "r"(p) :);
869 }
870
871 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p)
872 {
873     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(14), "r"(p) :);
874 }
875
876 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p)
877 {
878     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(15), "r"(p) :);
879 }
880
881 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p)
882 {
883     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(16), "r"(p) :);
884 }
885
886 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p)
887 {
888     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(17), "r"(p) :);
889 }
890
891 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p)
892 {
893     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(18), "r"(p) :);
894 }
895
896 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p)
897 {
898     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(19), "r"(p) :);
899 }
900
901 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p)
```

```
902 {
903     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(20), "r"(p) :);
904 }
905
906 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p)
907 {
908     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(21), "r"(p) :);
909 }
910
911 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p)
912 {
913     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(22), "r"(p) :);
914 }
915
916 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p)
917 {
918     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(23), "r"(p) :);
919 }
920
921 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p)
922 {
923     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(24), "r"(p) :);
924 }
925
926 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p)
927 {
928     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(25), "r"(p) :);
929 }
930
931 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p)
932 {
933     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(26), "r"(p) :);
934 }
935
936 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p)
937 {
938     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(27), "r"(p) :);
939 }
940
941 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p)
942 {
943     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(28), "r"(p) :);
944 }
945
946 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p)
947 {
948     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(29), "r"(p) :);
949 }
950
951 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p)
952 {
953     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(30), "r"(p) :);
954 }
955
956 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p)
957 {
958     asm volatile("call _optix_set_payload, (%0, %1);" : : "r"(31), "r"(p) :);
959 }
960
961 static __forceinline__ __device__ unsigned int optixGetPayload_0()
962 {
963     unsigned int result;
964     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(0) :);
965     return result;
966 }
967
968 static __forceinline__ __device__ unsigned int optixGetPayload_1()
```

```
969 {
970     unsigned int result;
971     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(1) :);
972     return result;
973 }
974
975 static __forceinline__ __device__ unsigned int optixGetPayload_2()
976 {
977     unsigned int result;
978     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(2) :);
979     return result;
980 }
981
982 static __forceinline__ __device__ unsigned int optixGetPayload_3()
983 {
984     unsigned int result;
985     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(3) :);
986     return result;
987 }
988
989 static __forceinline__ __device__ unsigned int optixGetPayload_4()
990 {
991     unsigned int result;
992     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(4) :);
993     return result;
994 }
995
996 static __forceinline__ __device__ unsigned int optixGetPayload_5()
997 {
998     unsigned int result;
999     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(5) :);
1000     return result;
1001 }
1002
1003 static __forceinline__ __device__ unsigned int optixGetPayload_6()
1004 {
1005     unsigned int result;
1006     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(6) :);
1007     return result;
1008 }
1009
1010 static __forceinline__ __device__ unsigned int optixGetPayload_7()
1011 {
1012     unsigned int result;
1013     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(7) :);
1014     return result;
1015 }
1016
1017 static __forceinline__ __device__ unsigned int optixGetPayload_8()
1018 {
1019     unsigned int result;
1020     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(8) :);
1021     return result;
1022 }
1023
1024 static __forceinline__ __device__ unsigned int optixGetPayload_9()
1025 {
1026     unsigned int result;
1027     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(9) :);
1028     return result;
1029 }
1030
1031 static __forceinline__ __device__ unsigned int optixGetPayload_10()
1032 {
1033     unsigned int result;
1034     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(10) :);
1035     return result;
```

```
1036 }
1037
1038 static __forceinline__ __device__ unsigned int optixGetPayload_11()
1039 {
1040     unsigned int result;
1041     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(11) :);
1042     return result;
1043 }
1044
1045 static __forceinline__ __device__ unsigned int optixGetPayload_12()
1046 {
1047     unsigned int result;
1048     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(12) :);
1049     return result;
1050 }
1051
1052 static __forceinline__ __device__ unsigned int optixGetPayload_13()
1053 {
1054     unsigned int result;
1055     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(13) :);
1056     return result;
1057 }
1058
1059 static __forceinline__ __device__ unsigned int optixGetPayload_14()
1060 {
1061     unsigned int result;
1062     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(14) :);
1063     return result;
1064 }
1065
1066 static __forceinline__ __device__ unsigned int optixGetPayload_15()
1067 {
1068     unsigned int result;
1069     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(15) :);
1070     return result;
1071 }
1072
1073 static __forceinline__ __device__ unsigned int optixGetPayload_16()
1074 {
1075     unsigned int result;
1076     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(16) :);
1077     return result;
1078 }
1079
1080 static __forceinline__ __device__ unsigned int optixGetPayload_17()
1081 {
1082     unsigned int result;
1083     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(17) :);
1084     return result;
1085 }
1086
1087 static __forceinline__ __device__ unsigned int optixGetPayload_18()
1088 {
1089     unsigned int result;
1090     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(18) :);
1091     return result;
1092 }
1093
1094 static __forceinline__ __device__ unsigned int optixGetPayload_19()
1095 {
1096     unsigned int result;
1097     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(19) :);
1098     return result;
1099 }
1100
1101 static __forceinline__ __device__ unsigned int optixGetPayload_20()
1102 {
```

```
1103     unsigned int result;
1104     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(20) :);
1105     return result;
1106 }
1107
1108 static __forceinline__ __device__ unsigned int optixGetPayload_21()
1109 {
1110     unsigned int result;
1111     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(21) :);
1112     return result;
1113 }
1114
1115 static __forceinline__ __device__ unsigned int optixGetPayload_22()
1116 {
1117     unsigned int result;
1118     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(22) :);
1119     return result;
1120 }
1121
1122 static __forceinline__ __device__ unsigned int optixGetPayload_23()
1123 {
1124     unsigned int result;
1125     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(23) :);
1126     return result;
1127 }
1128
1129 static __forceinline__ __device__ unsigned int optixGetPayload_24()
1130 {
1131     unsigned int result;
1132     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(24) :);
1133     return result;
1134 }
1135
1136 static __forceinline__ __device__ unsigned int optixGetPayload_25()
1137 {
1138     unsigned int result;
1139     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(25) :);
1140     return result;
1141 }
1142
1143 static __forceinline__ __device__ unsigned int optixGetPayload_26()
1144 {
1145     unsigned int result;
1146     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(26) :);
1147     return result;
1148 }
1149
1150 static __forceinline__ __device__ unsigned int optixGetPayload_27()
1151 {
1152     unsigned int result;
1153     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(27) :);
1154     return result;
1155 }
1156
1157 static __forceinline__ __device__ unsigned int optixGetPayload_28()
1158 {
1159     unsigned int result;
1160     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(28) :);
1161     return result;
1162 }
1163
1164 static __forceinline__ __device__ unsigned int optixGetPayload_29()
1165 {
1166     unsigned int result;
1167     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(29) :);
1168     return result;
1169 }
```

```
1170
1171 static __forceinline__ __device__ unsigned int optixGetPayload_30()
1172 {
1173     unsigned int result;
1174     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(30) :);
1175     return result;
1176 }
1177
1178 static __forceinline__ __device__ unsigned int optixGetPayload_31()
1179 {
1180     unsigned int result;
1181     asm volatile("call (%0), _optix_get_payload, (%1);" : "=r"(result) : "r"(31) :);
1182     return result;
1183 }
1184
1185 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int types)
1186 {
1187     asm volatile("call _optix_set_payload_types, (%0);" : : "r"(types) :);
1188 }
1189
1190 static __forceinline__ __device__ unsigned int optixUndefinedValue()
1191 {
1192     unsigned int u0;
1193     asm("call (%0), _optix_undef_value, ();" : "=r"(u0) :);
1194     return u0;
1195 }
1196
1197 static __forceinline__ __device__ float3 optixGetWorldRayOrigin()
1198 {
1199     float f0, f1, f2;
1200     asm("call (%0), _optix_get_world_ray_origin_x, ();" : "=f"(f0) :);
1201     asm("call (%0), _optix_get_world_ray_origin_y, ();" : "=f"(f1) :);
1202     asm("call (%0), _optix_get_world_ray_origin_z, ();" : "=f"(f2) :);
1203     return make_float3(f0, f1, f2);
1204 }
1205
1206 static __forceinline__ __device__ float3 optixGetWorldRayDirection()
1207 {
1208     float f0, f1, f2;
1209     asm("call (%0), _optix_get_world_ray_direction_x, ();" : "=f"(f0) :);
1210     asm("call (%0), _optix_get_world_ray_direction_y, ();" : "=f"(f1) :);
1211     asm("call (%0), _optix_get_world_ray_direction_z, ();" : "=f"(f2) :);
1212     return make_float3(f0, f1, f2);
1213 }
1214
1215 static __forceinline__ __device__ float3 optixGetObjectRayOrigin()
1216 {
1217     float f0, f1, f2;
1218     asm("call (%0), _optix_get_object_ray_origin_x, ();" : "=f"(f0) :);
1219     asm("call (%0), _optix_get_object_ray_origin_y, ();" : "=f"(f1) :);
1220     asm("call (%0), _optix_get_object_ray_origin_z, ();" : "=f"(f2) :);
1221     return make_float3(f0, f1, f2);
1222 }
1223
1224 static __forceinline__ __device__ float3 optixGetObjectRayDirection()
1225 {
1226     float f0, f1, f2;
1227     asm("call (%0), _optix_get_object_ray_direction_x, ();" : "=f"(f0) :);
1228     asm("call (%0), _optix_get_object_ray_direction_y, ();" : "=f"(f1) :);
1229     asm("call (%0), _optix_get_object_ray_direction_z, ();" : "=f"(f2) :);
1230     return make_float3(f0, f1, f2);
1231 }
1232
1233 static __forceinline__ __device__ float optixGetRayTmin()
1234 {
1235     float f0;
1236     asm("call (%0), _optix_get_ray_tmin, ();" : "=f"(f0) :);
```

```
1237     return f0;
1238 }
1239
1240 static __forceinline__ __device__ float optixGetRayTmax()
1241 {
1242     float f0;
1243     asm("call (%0), _optix_get_ray_tmax, ();" : "=f"(f0) :);
1244     return f0;
1245 }
1246
1247 static __forceinline__ __device__ float optixGetRayTime()
1248 {
1249     float f0;
1250     asm("call (%0), _optix_get_ray_time, ();" : "=f"(f0) :);
1251     return f0;
1252 }
1253
1254 static __forceinline__ __device__ unsigned int optixGetRayFlags()
1255 {
1256     unsigned int u0;
1257     asm("call (%0), _optix_get_ray_flags, ();" : "=r"(u0) :);
1258     return u0;
1259 }
1260
1261 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask()
1262 {
1263     unsigned int u0;
1264     asm("call (%0), _optix_get_ray_visibility_mask, ();" : "=r"(u0) :);
1265     return u0;
1266 }
1267
1268 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias,
1269                                                                                 unsigned int
instIdx)
1270 {
1271     unsigned long long handle;
1272     asm("call (%0), _optix_get_instance_traversable_from_ias, (%1, %2);"
1273         : "=l"(handle) : "l"(ias), "r"(instIdx));
1274     return (OptixTraversableHandle)handle;
1275 }
1276
1277
1278 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas,
1279                                                                    unsigned int           primIdx,
1280                                                                    unsigned int           sbtGASIndex,
1281                                                                    float                  time,
1282                                                                    float3                 data[3])
1283 {
1284     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_triangle_vertex_data, "
1285         "(%9, %10, %11, %12);"
1286         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
1287           "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1288         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1289         :);
1290 }
1291
1292
1293 static __forceinline__ __device__ void optixGetTriangleVertexDataFromHandle(OptixTraversableHandle gas,
1294                                                                              unsigned int           primIdx,
1295                                                                            unsigned int           sbtGASIndex,
1296                                                                                 float                  time,
1297                                                                               float3                 data[3])
1298 {
1299     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_triangle_vertex_data_from_handle, "
1300         "(%9, %10, %11, %12);"
1301         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
```

```
1302             "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1303           : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1304           :);
1305 }
1306
1307 static __forceinline__ __device__ void optixGetTriangleVertexData(float3 data[3])
1308 {
1309     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_triangle_vertex_data_current_hit, "
1310         "();"
1311         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
1312           "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1313         :);
1314 }
1315
1316 static __forceinline__ __device__ void optixHitObjectGetTriangleVertexData(float3 data[3])
1317 {
1318     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_hitobject_get_triangle_vertex_data, "
1319         "();"
1320         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[1].x), "=f"(data[1].y),
1321           "=f"(data[1].z), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z)
1322         :);
1323 }
1324
1325
1326 static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
1327                                                                       unsigned int           primIdx,
1328                                                                     unsigned int           sbtGASIndex,
1329                                                                       float                  time,
1330                                                                       float4                 data[2])
1331 {
1332     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7), _optix_get_linear_curve_vertex_data, "
1333         "(%8, %9, %10, %11);"
1334         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1335           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
1336         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1337         :);
1338 }
1339
1340 static __forceinline__ __device__ void optixGetLinearCurveVertexDataFromHandle(OptixTraversableHandle
gas,
1341                                                                                 unsigned int          primIdx,
1342                                                                                       unsigned int
sbtGASIndex,
1343                                                                                    float                time,
1344                                                                                  float4                 data[2])
1345 {
1346     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7), _optix_get_linear_curve_vertex_data_from_handle, "
1347         "(%8, %9, %10, %11);"
1348         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1349           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
1350         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1351         :);
1352 }
1353
1354 static __forceinline__ __device__ void optixGetLinearCurveVertexData(float4 data[2])
1355 {
1356     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7), _optix_get_linear_curve_vertex_data_current_hit, "
1357         "();"
1358         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1359           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
1360         :);
1361 }
1362
1363 static __forceinline__ __device__ void optixHitObjectGetLinearCurveVertexData(float4 data[2])
1364
1365 {
1366     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7), _optix_hitobject_get_linear_curve_vertex_data, "
```

```
1367          "();"
1368          : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1369            "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w)
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
optixGetQuadraticBSplineVertexDataFromHandle(OptixTraversableHandle gas,
1389                                                                          unsigned int         primIdx,
1390                                                                                 unsigned int
sbtGASIndex,
1391                                                                            float                time,
1392                                                                          float4               data[3])
1393 {
1394     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11),
_optix_get_quadratic_bspline_vertex_data_from_handle, "
1395          "(%12, %13, %14, %15);"
1396          : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1397            "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1398            "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w)
1399          : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1400          :);
1401 }
1402
1403 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(float4 data[3])
1404 {
1405     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11),
_optix_get_quadratic_bspline_vertex_data_current_hit, "
1406          "();"
1407          : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1408            "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1409            "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w)
1410          :);
1411 }
1412
1413 static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineVertexData(float4 data[3])
1414 {
1415     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11),
_optix_hitobject_get_quadratic_bspline_vertex_data, "
1416          "();"
1417          : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1418            "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1419            "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w)
1420          :);
1421 }
1422
1423 static __forceinline__ __device__ void
optixGetQuadraticBSplineRocapsVertexDataFromHandle(OptixTraversableHandle gas,
1424                                                                                unsigned int primIdx,
1425                                                                                        unsigned int
sbtGASIndex,
```

```
1426                                                                                    float        time,
1427                                                                                  float4       data[3])
1428 {
1429     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11),
_optix_get_quadratic_bspline_rocaps_vertex_data_from_handle, "
1430         "(%12, %13, %14, %15);"
1431         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
"=f"(data[1].y),
1432           "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
"=f"(data[2].w)
1433         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1434         :);
1435 }
1436
1437 static __forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexData(float4 data[3])
1438 {
1439     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11),
_optix_get_quadratic_bspline_rocaps_vertex_data_current_hit, "
1440         "();"
1441         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
"=f"(data[1].y),
1442           "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
"=f"(data[2].w)
1443         :);
1444 }
1445
1446 static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineRocapsVertexData(float4 data[3])
1447 {
1448     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11),
_optix_hitobject_get_quadratic_bspline_rocaps_vertex_data, "
1449         "();"
1450         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
"=f"(data[1].y),
1451           "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
"=f"(data[2].w)
1452         :);
1453 }
1454
1455 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,
1456                                                                        unsigned int         primIdx,
1457                                                                        unsigned int         sbtGASIndex,
1458                                                                        float                time,
1459                                                                        float4               data[4])
1460 {
1461     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1462         "_optix_get_cubic_bspline_vertex_data, "
1463         "(%16, %17, %18, %19);"
1464         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1465           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1466           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
1467           "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
1468         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1469         :);
1470 }
1471
1472 static __forceinline__ __device__ void optixGetCubicBSplineVertexDataFromHandle(OptixTraversableHandle
gas,
1473                                                                                  unsigned int         primIdx,
1474                                                                                unsigned int         sbtGASIndex,
1475                                                                                      float                time,
1476                                                                                    float4               data[4])
1477 {
1478     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1479         "_optix_get_cubic_bspline_vertex_data_from_handle, "
1480         "(%16, %17, %18, %19);"
1481         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1482           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
```

```
1483             "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
1484             "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
1485           : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1486           :);
1487 }
1488
1489 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(float4 data[4])
1490 {
1491     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1492         "_optix_get_cubic_bspline_vertex_data_current_hit, "
1493         "();"
1494         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1495           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1496           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
1497           "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
1498         :);
1499 }
1500
1501 static __forceinline__ __device__ void optixHitObjectGetCubicBSplineVertexData(float4 data[4])
1502 {
1503     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1504         "_optix_hitobject_get_cubic_bspline_vertex_data, "
1505         "();"
1506         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w),
1507           "=f"(data[1].x), "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w),
1508           "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z), "=f"(data[2].w),
1509           "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z), "=f"(data[3].w)
1510         :);
1511 }
1512
1513 static __forceinline__ __device__ void
optixGetCubicBSplineRocapsVertexDataFromHandle(OptixTraversableHandle gas,
1514                                                                                 unsigned int
primIdx,
1515                                                                               unsigned int sbtGASIndex,
1516                                                                                     float        time,
1517                                                                                 float4       data[4])
1518 {
1519     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1520         "_optix_get_cubic_bspline_rocaps_vertex_data_from_handle, "
1521         "(%16, %17, %18, %19);"
1522         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1523           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1524           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1525         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1526         :);
1527 }
1528
1529 static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexData(float4 data[4])
1530 {
1531     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1532         "_optix_get_cubic_bspline_rocaps_vertex_data_current_hit, "
1533         "();"
1534         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1535           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1536           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1537         :);
1538 }
1539
1540 static __forceinline__ __device__ void optixHitObjectGetCubicBSplineRocapsVertexData(float4 data[4])
1541 {
1542     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1543         "_optix_hitobject_get_cubic_bspline_rocaps_vertex_data, "
1544         "();"
1545         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
```

```
1546            "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1547            "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1548          :);
1549 }
1550
1551 static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas,
1552                                                                      unsigned int           primIdx,
1553                                                                     unsigned int           sbtGASIndex,
1554                                                                      float                  time,
1555                                                                      float4                 data[4])
1556 {
1557     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1558         "_optix_get_catmullrom_vertex_data, "
1559         "(%16, %17, %18, %19);"
1560         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1561           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1562           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1563         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1564         :);
1565 }
1566
1567 static __forceinline__ __device__ void optixGetCatmullRomVertexDataFromHandle(OptixTraversableHandle
gas,
1568                                                                                unsigned int           primIdx,
1569                                                                               unsigned int           sbtGASIndex,
1570                                                                                 float                  time,
1571                                                                                float4                 data[4])
1572 {
1573     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1574         "_optix_get_catmullrom_vertex_data_from_handle, "
1575         "(%16, %17, %18, %19);"
1576         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1577           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1578           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1579         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1580         :);
1581 }
1582
1583 static __forceinline__ __device__ void optixGetCatmullRomVertexData(float4 data[4])
1584 {
1585     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1586         "_optix_get_catmullrom_vertex_data_current_hit, "
1587         "();"
1588         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1589           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1590           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1591         :);
1592 }
1593
1594 static __forceinline__ __device__ void optixHitObjectGetCatmullRomVertexData(float4 data[4])
1595 {
1596     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1597         "_optix_hitobject_get_catmullrom_vertex_data, "
1598         "();"
1599         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1600           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1601           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1602         :);
1603 }
1604
1605 static __forceinline__ __device__ void
optixGetCatmullRomRocapsVertexDataFromHandle(OptixTraversableHandle gas,
```

```
1606                                                                    unsigned int
primIdx,
1607                                                                    unsigned int
sbtGASIndex,
1608                                                                 float                time,
1609                                                                    float4
data[4])
1610 {
1611     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1612         "_optix_get_catmullrom_rocaps_vertex_data_from_handle, "
1613         "(%16, %17, %18, %19);"
1614         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1615           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1616           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1617         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1618         :);
1619 }
1620
1621 static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexData(float4 data[4])
1622 {
1623     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1624         "_optix_get_catmullrom_rocaps_vertex_data_current_hit, "
1625         "();"
1626         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1627           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1628           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1629         :);
1630 }
1631
1632 static __forceinline__ __device__ void optixHitObjectGetCatmullRomRocapsVertexData(float4 data[4])
1633 {
1634     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1635         "_optix_hitobject_get_catmullrom_rocaps_vertex_data, "
1636         "();"
1637         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1638           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1639           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1640         :);
1641 }
1642
1643 static __forceinline__ __device__ void optixGetCubicBezierVertexData(OptixTraversableHandle gas,
1644                                                                       unsigned int           primIdx,
1645                                                                     unsigned int           sbtGASIndex,
1646                                                                       float                time,
1647                                                                       float4               data[4])
1648 {
1649     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1650         "_optix_get_cubic_bezier_vertex_data, "
1651         "(%16, %17, %18, %19);"
1652         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1653           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1654           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1655         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1656         :);
1657 }
1658
1659 static __forceinline__ __device__ void optixGetCubicBezierVertexDataFromHandle(OptixTraversableHandle
gas,
1660                                                                         unsigned int           primIdx,
1661                                                                       unsigned int           sbtGASIndex,
1662                                                                           float                time,
1663                                                                         float4               data[4])
1664 {
```

```
1665     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1666         "_optix_get_cubic_bezier_vertex_data_from_handle, "
1667         "(%16, %17, %18, %19);"
1668         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1669           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1670           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1671         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1672         :);
1673 }
1674
1675 static __forceinline__ __device__ void optixGetCubicBezierVertexData(float4 data[4])
1676 {
1677     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1678         "_optix_get_cubic_bezier_vertex_data_current_hit, "
1679         "();"
1680         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1681           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1682           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1683         :);
1684 }
1685
1686 static __forceinline__ __device__ void optixHitObjectGetCubicBezierVertexData(float4 data[4])
1687 {
1688     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1689         "_optix_hitobject_get_cubic_bezier_vertex_data, "
1690         "();"
1691         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1692           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1693           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1694         :);
1695 }
1696
1697 static __forceinline__ __device__ void
optixGetCubicBezierRocapsVertexDataFromHandle(OptixTraversableHandle gas,
1698                                                                                 unsigned int
primIdx,
1699                                                                               unsigned int sbtGASIndex,
1700                                                                                     float        time,
1701                                                                                 float4       data[4])
1702 {
1703     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1704         "_optix_get_cubic_bezier_rocaps_vertex_data_from_handle, "
1705         "(%16, %17, %18, %19);"
1706         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1707           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1708           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1709         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1710         :);
1711 }
1712
1713 static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexData(float4 data[4])
1714 {
1715     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1716         "_optix_get_cubic_bezier_rocaps_vertex_data_current_hit, "
1717         "();"
1718         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1719           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1720           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1721         :);
1722 }
1723
1724 static __forceinline__ __device__ void optixHitObjectGetCubicBezierRocapsVertexData(float4 data[4])
```

```
1725 {
1726     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11,  %12, %13, %14, %15), "
1727         "_optix_hitobject_get_cubic_bezier_rocaps_vertex_data, "
1728         "();"
1729         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
1730           "=f"(data[1].y), "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y),
1731           "=f"(data[2].z), "=f"(data[2].w), "=f"(data[3].x), "=f"(data[3].y), "=f"(data[3].z),
"=f"(data[3].w)
1732         :);
1733 }
1734
1735 static __forceinline__ __device__ void optixGetRibbonVertexData(OptixTraversableHandle gas,
1736                                                                  unsigned int           primIdx,
1737                                                                  unsigned int           sbtGASIndex,
1738                                                                  float                  time,
1739                                                                  float4                 data[3])
1740 {
1741     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11), _optix_get_ribbon_vertex_data, "
1742         "(%12, %13, %14, %15);"
1743         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
"=f"(data[1].y),
1744           "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
"=f"(data[2].w)
1745         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1746         :);
1747 }
1748
1749 static __forceinline__ __device__ void optixGetRibbonVertexDataFromHandle(OptixTraversableHandle gas,
1750                                                                            unsigned int           primIdx,
1751                                                                          unsigned int           sbtGASIndex,
1752                                                                            float                  time,
1753                                                                          float4                 data[3])
1754 {
1755     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11),
_optix_get_ribbon_vertex_data_from_handle, "
1756         "(%12, %13, %14, %15);"
1757         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
"=f"(data[1].y),
1758           "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
"=f"(data[2].w)
1759         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1760         :);
1761 }
1762
1763 static __forceinline__ __device__ void optixGetRibbonVertexData(float4 data[3])
1764 {
1765     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11),
_optix_get_ribbon_vertex_data_current_hit, "
1766         "();"
1767         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
"=f"(data[1].y),
1768           "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
"=f"(data[2].w)
1769         :);
1770 }
1771
1772 static __forceinline__ __device__ void optixHitObjectGetRibbonVertexData(float4 data[3])
1773 {
1774     asm("call (%0, %1, %2, %3,  %4, %5, %6, %7,  %8, %9, %10, %11),
_optix_hitobject_get_ribbon_vertex_data, "
1775         "();"
1776         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w), "=f"(data[1].x),
"=f"(data[1].y),
1777           "=f"(data[1].z), "=f"(data[1].w), "=f"(data[2].x), "=f"(data[2].y), "=f"(data[2].z),
"=f"(data[2].w)
1778         :);
1779 }
```

```
1780
1781 static __forceinline__ __device__ float3 optixGetRibbonNormal(OptixTraversableHandle gas,
1782                                                               unsigned int           primIdx,
1783                                                               unsigned int           sbtGASIndex,
1784                                                               float                  time,
1785                                                               float2                 ribbonParameters)
1786 {
1787     float3 normal;
1788     asm("call (%0, %1, %2), _optix_get_ribbon_normal, "
1789         "(%3, %4, %5, %6, %7, %8);"
1790         : "=f"(normal.x), "=f"(normal.y), "=f"(normal.z)
1791         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time),
1792           "f"(ribbonParameters.x), "f"(ribbonParameters.y)
1793         :);
1794     return normal;
1795 }
1796
1797 static __forceinline__ __device__ float3 optixGetRibbonNormalFromHandle(OptixTraversableHandle gas,
1798                                                                         unsigned int           primIdx,
1799                                                                     unsigned int           sbtGASIndex,
1800                                                                         float                  time,
1801                                                                   float2                 ribbonParameters)
1802 {
1803     float3 normal;
1804     asm("call (%0, %1, %2), _optix_get_ribbon_normal_from_handle, "
1805         "(%3, %4, %5, %6, %7, %8);"
1806         : "=f"(normal.x), "=f"(normal.y), "=f"(normal.z)
1807         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time),
1808           "f"(ribbonParameters.x), "f"(ribbonParameters.y)
1809         :);
1810     return normal;
1811 }
1812
1813 static __forceinline__ __device__ float3 optixGetRibbonNormal(float2 ribbonParameters)
1814 {
1815     float3 normal;
1816     asm("call (%0, %1, %2), _optix_get_ribbon_normal_current_hit, "
1817         "(%3, %4);"
1818         : "=f"(normal.x), "=f"(normal.y), "=f"(normal.z)
1819         : "f"(ribbonParameters.x), "f"(ribbonParameters.y)
1820         :);
1821     return normal;
1822 }
1823
1824 static __forceinline__ __device__ float3 optixHitObjectGetRibbonNormal(float2 ribbonParameters)
1825 {
1826     float3 normal;
1827     asm("call (%0, %1, %2), _optix_hitobject_get_ribbon_normal, "
1828         "(%3, %4);"
1829         : "=f"(normal.x), "=f"(normal.y), "=f"(normal.z)
1830         : "f"(ribbonParameters.x), "f"(ribbonParameters.y)
1831         :);
1832     return normal;
1833 }
1834
1835 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas,
1836                                                           unsigned int           primIdx,
1837                                                           unsigned int           sbtGASIndex,
1838                                                           float                  time,
1839                                                           float4                 data[1])
1840 {
1841     asm("call (%0, %1, %2, %3), "
1842         "_optix_get_sphere_data, "
1843         "(%4, %5, %6, %7);"
1844         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
1845         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1846         :);
```

```
1847 }
1848
1849 static __forceinline__ __device__ void optixGetSphereDataFromHandle(OptixTraversableHandle gas,
1850                                                                      unsigned int           primIdx,
1851                                                                     unsigned int           sbtGASIndex,
1852                                                                      float                  time,
1853                                                                      float4                 data[1])
1854 {
1855     asm("call (%0, %1, %2, %3), "
1856         "_optix_get_sphere_data_from_handle, "
1857         "(%4, %5, %6, %7);"
1858         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
1859         : "l"(gas), "r"(primIdx), "r"(sbtGASIndex), "f"(time)
1860         :);
1861 }
1862
1863 static __forceinline__ __device__ void optixGetSphereData(float4 data[1])
1864 {
1865     asm("call (%0, %1, %2, %3), "
1866         "_optix_get_sphere_data_current_hit, "
1867         "();"
1868         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
1869         :);
1870 }
1871
1872 static __forceinline__ __device__ void optixHitObjectGetSphereData(float4 data[1])
1873 {
1874     asm("call (%0, %1, %2, %3), "
1875         "_optix_hitobject_get_sphere_data, "
1876         "();"
1877         : "=f"(data[0].x), "=f"(data[0].y), "=f"(data[0].z), "=f"(data[0].w)
1878         :);
1879 }
1880
1881 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle()
1882 {
1883     unsigned long long handle;
1884     asm("call (%0), _optix_get_gas_traversable_handle, ();" : "=l"(handle) :);
1885     return (OptixTraversableHandle)handle;
1886 }
1887
1888 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle handle)
1889 {
1890     float f0;
1891     asm("call (%0), _optix_get_gas_motion_time_begin, (%1);" : "=f"(f0) : "l"(handle) :);
1892     return f0;
1893 }
1894
1895 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle handle)
1896 {
1897     float f0;
1898     asm("call (%0), _optix_get_gas_motion_time_end, (%1);" : "=f"(f0) : "l"(handle) :);
1899     return f0;
1900 }
1901
1902 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle handle)
1903 {
1904     unsigned int u0;
1905     asm("call (%0), _optix_get_gas_motion_step_count, (%1);" : "=r"(u0) : "l"(handle) :);
1906     return u0;
1907 }
1908
1909 template<typename HitState>
1910 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(const HitState& hs, float
m[12])
1911 {
1912     if(hs.getTransformListSize() == 0)
```

```
1913     {
1914         m[0]  = 1.0f;
1915         m[1]  = 0.0f;
1916         m[2]  = 0.0f;
1917         m[3]  = 0.0f;
1918         m[4]  = 0.0f;
1919         m[5]  = 1.0f;
1920         m[6]  = 0.0f;
1921         m[7]  = 0.0f;
1922         m[8]  = 0.0f;
1923         m[9]  = 0.0f;
1924         m[10] = 1.0f;
1925         m[11] = 0.0f;
1926         return;
1927     }
1928
1929     float4 m0, m1, m2;
1930     optix_impl::optixGetWorldToObjectTransformMatrix(hs, m0, m1, m2);
1931     m[0]  = m0.x;
1932     m[1]  = m0.y;
1933     m[2]  = m0.z;
1934     m[3]  = m0.w;
1935     m[4]  = m1.x;
1936     m[5]  = m1.y;
1937     m[6]  = m1.z;
1938     m[7]  = m1.w;
1939     m[8]  = m2.x;
1940     m[9]  = m2.y;
1941     m[10] = m2.z;
1942     m[11] = m2.w;
1943 }
1944
1945 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12])
1946 {
1947     optixGetWorldToObjectTransformMatrix(OptixIncomingHitObject{}, m);
1948 }
1949
1950 static __forceinline__ __device__ void optixHitObjectGetWorldToObjectTransformMatrix(float m[12])
1951 {
1952     optixGetWorldToObjectTransformMatrix(OptixOutgoingHitObject{}, m);
1953 }
1954
1955 template<typename HitState>
1956 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(const HitState& hs, float
m[12])
1957 {
1958     if(hs.getTransformListSize() == 0)
1959     {
1960         m[0]  = 1.0f;
1961         m[1]  = 0.0f;
1962         m[2]  = 0.0f;
1963         m[3]  = 0.0f;
1964         m[4]  = 0.0f;
1965         m[5]  = 1.0f;
1966         m[6]  = 0.0f;
1967         m[7]  = 0.0f;
1968         m[8]  = 0.0f;
1969         m[9]  = 0.0f;
1970         m[10] = 1.0f;
1971         m[11] = 0.0f;
1972         return;
1973     }
1974
1975     float4 m0, m1, m2;
1976     optix_impl::optixGetObjectToWorldTransformMatrix(hs, m0, m1, m2);
1977     m[0]  = m0.x;
1978     m[1]  = m0.y;
```

```
1979     m[2]  = m0.z;
1980     m[3]  = m0.w;
1981     m[4]  = m1.x;
1982     m[5]  = m1.y;
1983     m[6]  = m1.z;
1984     m[7]  = m1.w;
1985     m[8]  = m2.x;
1986     m[9]  = m2.y;
1987     m[10] = m2.z;
1988     m[11] = m2.w;
1989 }
1990
1991 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12])
1992 {
1993     optixGetObjectToWorldTransformMatrix(OptixIncomingHitObject{}, m);
1994 }
1995
1996 static __forceinline__ __device__ void optixHitObjectGetObjectToWorldTransformMatrix(float m[12])
1997 {
1998     optixGetObjectToWorldTransformMatrix(OptixOutgoingHitObject{}, m);
1999 }
2000
2001 template<typename HitState>
2002 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(const HitState& hs,
float3 point)
2003 {
2004     if(hs.getTransformListSize() == 0)
2005         return point;
2006
2007     float4 m0, m1, m2;
2008     optix_impl::optixGetWorldToObjectTransformMatrix(hs, m0, m1, m2);
2009     return optix_impl::optixTransformPoint(m0, m1, m2, point);
2010 }
2011
2012 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point)
2013 {
2014     return optixTransformPointFromWorldToObjectSpace(OptixIncomingHitObject{}, point);
2015 }
2016
2017 static __forceinline__ __device__ float3 optixHitObjectTransformPointFromWorldToObjectSpace(float3
point)
2018 {
2019     return optixTransformPointFromWorldToObjectSpace(OptixOutgoingHitObject{}, point);
2020 }
2021
2022 template<typename HitState>
2023 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(const HitState& hs,
float3 vec)
2024 {
2025     if(hs.getTransformListSize() == 0)
2026         return vec;
2027
2028     float4 m0, m1, m2;
2029     optix_impl::optixGetWorldToObjectTransformMatrix(hs, m0, m1, m2);
2030     return optix_impl::optixTransformVector(m0, m1, m2, vec);
2031 }
2032
2033 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec)
2034 {
2035     return optixTransformVectorFromWorldToObjectSpace(OptixIncomingHitObject{}, vec);
2036 }
2037
2038 static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromWorldToObjectSpace(float3 vec)
2039 {
2040     return optixTransformVectorFromWorldToObjectSpace(OptixOutgoingHitObject{}, vec);
2041 }
2042
```

```
2043 template<typename HitState>
2044 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(const HitState& hs,
float3 normal)
2045 {
2046     if(hs.getTransformListSize() == 0)
2047         return normal;
2048
2049     float4 m0, m1, m2;
2050     optix_impl::optixGetObjectToWorldTransformMatrix(hs, m0, m1, m2);  // inverse of
optixGetWorldToObjectTransformMatrix()
2051     return optix_impl::optixTransformNormal(m0, m1, m2, normal);
2052 }
2053
2054 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal)
2055 {
2056     return optixTransformNormalFromWorldToObjectSpace(OptixIncomingHitObject{}, normal);
2057 }
2058
2059 static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromWorldToObjectSpace(float3
normal)
2060 {
2061     return optixTransformNormalFromWorldToObjectSpace(OptixOutgoingHitObject{}, normal);
2062 }
2063
2064 template<typename HitState>
2065 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(const HitState& hs,
float3 point)
2066 {
2067     if(hs.getTransformListSize() == 0)
2068         return point;
2069
2070     float4 m0, m1, m2;
2071     optix_impl::optixGetObjectToWorldTransformMatrix(hs, m0, m1, m2);
2072     return optix_impl::optixTransformPoint(m0, m1, m2, point);
2073 }
2074
2075 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point)
2076 {
2077     return optixTransformPointFromObjectToWorldSpace(OptixIncomingHitObject{}, point);
2078 }
2079
2080 static __forceinline__ __device__ float3 optixHitObjectTransformPointFromObjectToWorldSpace(float3
point)
2081 {
2082     return optixTransformPointFromObjectToWorldSpace(OptixOutgoingHitObject{}, point);
2083 }
2084
2085 template<typename HitState>
2086 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(const HitState& hs,
float3 vec)
2087 {
2088     if(hs.getTransformListSize() == 0)
2089         return vec;
2090
2091     float4 m0, m1, m2;
2092     optix_impl::optixGetObjectToWorldTransformMatrix(hs, m0, m1, m2);
2093     return optix_impl::optixTransformVector(m0, m1, m2, vec);
2094 }
2095
2096 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec)
2097 {
2098     return optixTransformVectorFromObjectToWorldSpace(OptixIncomingHitObject{}, vec);
2099 }
2100
2101 static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromObjectToWorldSpace(float3 vec)
2102 {
2103     return optixTransformVectorFromObjectToWorldSpace(OptixOutgoingHitObject{}, vec);
```

```
2104 }
2105
2106 template<typename HitState>
2107 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(const HitState& hs,
float3 normal)
2108 {
2109     if(hs.getTransformListSize() == 0)
2110         return normal;
2111
2112     float4 m0, m1, m2;
2113     optix_impl::optixGetWorldToObjectTransformMatrix(hs, m0, m1, m2);  // inverse of
optixGetObjectToWorldTransformMatrix()
2114     return optix_impl::optixTransformNormal(m0, m1, m2, normal);
2115 }
2116
2117 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal)
2118 {
2119     return optixTransformNormalFromObjectToWorldSpace(OptixIncomingHitObject{}, normal);
2120 }
2121
2122 static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromObjectToWorldSpace(float3
normal)
2123 {
2124     return optixTransformNormalFromObjectToWorldSpace(OptixOutgoingHitObject{}, normal);
2125 }
2126
2127 static __forceinline__ __device__ unsigned int optixGetTransformListSize()
2128 {
2129     unsigned int u0;
2130     asm("call (%0), _optix_get_transform_list_size, ();" : "=r"(u0) :);
2131     return u0;
2132 }
2133
2134 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int index)
2135 {
2136     unsigned long long u0;
2137     asm("call (%0), _optix_get_transform_list_handle, (%1);" : "=l"(u0) : "r"(index) :);
2138     return u0;
2139 }
2140
2141 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle)
2142 {
2143     int i0;
2144     asm("call (%0), _optix_get_transform_type_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
2145     return (OptixTransformType)i0;
2146 }
2147
2148 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle)
2149 {
2150     unsigned long long ptr;
2151     asm("call (%0), _optix_get_static_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
2152     return (const OptixStaticTransform*)ptr;
2153 }
2154
2155 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle)
2156 {
2157     unsigned long long ptr;
2158     asm("call (%0), _optix_get_srt_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
2159     return (const OptixSRTMotionTransform*)ptr;
2160 }
2161
2162 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle)
2163 {
```

```
2164     unsigned long long ptr;
2165     asm("call (%0), _optix_get_matrix_motion_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
2166     return (const OptixMatrixMotionTransform*)ptr;
2167 }
2168
2169 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle)
2170 {
2171     int i0;
2172     asm("call (%0), _optix_get_instance_id_from_handle, (%1);" : "=r"(i0) : "l"(handle) :);
2173     return i0;
2174 }
2175
2176 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle(OptixTraversableHandle handle)
2177 {
2178     unsigned long long i0;
2179     asm("call (%0), _optix_get_instance_child_from_handle, (%1);" : "=l"(i0) : "l"(handle) :);
2180     return (OptixTraversableHandle)i0;
2181 }
2182
2183 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle)
2184 {
2185     unsigned long long ptr;
2186     asm("call (%0), _optix_get_instance_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
2187     return (const float4*)ptr;
2188 }
2189
2190 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle)
2191 {
2192     unsigned long long ptr;
2193     asm("call (%0), _optix_get_instance_inverse_transform_from_handle, (%1);" : "=l"(ptr) : "l"(handle)
:);
2194     return (const float4*)ptr;
2195 }
2196
2197 static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle(OptixTraversableHandle
handle)
2198 {
2199     unsigned long long ptr;
2200     asm("call (%0), _optix_get_gas_ptr_from_handle, (%1);" : "=l"(ptr) : "l"(handle) :);
2201     return (CUdeviceptr)ptr;
2202 }
2203 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind)
2204 {
2205     int ret;
2206     asm volatile(
2207         "call (%0), _optix_report_intersection_0"
2208         ", (%1, %2);"
2209         : "=r"(ret)
2210         : "f"(hitT), "r"(hitKind)
2211         :);
2212     return ret;
2213 }
2214
2215 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0)
2216 {
2217     int ret;
2218     asm volatile(
2219         "call (%0), _optix_report_intersection_1"
2220         ", (%1, %2, %3);"
2221         : "=r"(ret)
2222         : "f"(hitT), "r"(hitKind), "r"(a0)
2223         :);
```

```
2224     return ret;
2225 }
2226
2227 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1)
2228 {
2229     int ret;
2230     asm volatile(
2231         "call (%0), _optix_report_intersection_2"
2232         ", (%1, %2, %3, %4);"
2233         : "=r"(ret)
2234         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1)
2235         :);
2236     return ret;
2237 }
2238
2239 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1, unsigned int a2)
2240 {
2241     int ret;
2242     asm volatile(
2243         "call (%0), _optix_report_intersection_3"
2244         ", (%1, %2, %3, %4, %5);"
2245         : "=r"(ret)
2246         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2)
2247         :);
2248     return ret;
2249 }
2250
2251 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
2252                                                                unsigned int hitKind,
2253                                                                unsigned int a0,
2254                                                                unsigned int a1,
2255                                                                unsigned int a2,
2256                                                                unsigned int a3)
2257 {
2258     int ret;
2259     asm volatile(
2260         "call (%0), _optix_report_intersection_4"
2261         ", (%1, %2, %3, %4, %5, %6);"
2262         : "=r"(ret)
2263         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3)
2264         :);
2265     return ret;
2266 }
2267
2268 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
2269                                                                unsigned int hitKind,
2270                                                                unsigned int a0,
2271                                                                unsigned int a1,
2272                                                                unsigned int a2,
2273                                                                unsigned int a3,
2274                                                                unsigned int a4)
2275 {
2276     int ret;
2277     asm volatile(
2278         "call (%0), _optix_report_intersection_5"
2279         ", (%1, %2, %3, %4, %5, %6, %7);"
2280         : "=r"(ret)
2281         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4)
2282         :);
2283     return ret;
2284 }
2285
2286 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
2287                                                                unsigned int hitKind,
2288                                                                unsigned int a0,
```

```
2289                                                                unsigned int a1,
2290                                                                unsigned int a2,
2291                                                                unsigned int a3,
2292                                                                unsigned int a4,
2293                                                                unsigned int a5)
2294 {
2295     int ret;
2296     asm volatile(
2297         "call (%0), _optix_report_intersection_6"
2298         ", (%1, %2, %3, %4, %5, %6, %7, %8);"
2299         : "=r"(ret)
2300         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5)
2301         :);
2302     return ret;
2303 }
2304
2305 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
2306                                                                unsigned int hitKind,
2307                                                                unsigned int a0,
2308                                                                unsigned int a1,
2309                                                                unsigned int a2,
2310                                                                unsigned int a3,
2311                                                                unsigned int a4,
2312                                                                unsigned int a5,
2313                                                                unsigned int a6)
2314 {
2315     int ret;
2316     asm volatile(
2317         "call (%0), _optix_report_intersection_7"
2318         ", (%1, %2, %3, %4, %5, %6, %7, %8, %9);"
2319         : "=r"(ret)
2320         : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6)
2321         :);
2322     return ret;
2323 }
2324
2325 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
2326                                                                unsigned int hitKind,
2327                                                                unsigned int a0,
2328                                                                unsigned int a1,
2329                                                                unsigned int a2,
2330                                                                unsigned int a3,
2331                                                                unsigned int a4,
2332                                                                unsigned int a5,
2333                                                                unsigned int a6,
2334                                                                unsigned int a7)
2335 {
2336     int ret;
2337     asm volatile(
2338         "call (%0), _optix_report_intersection_8"
2339         ", (%1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
2340         : "=r"(ret)
2341        : "f"(hitT), "r"(hitKind), "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5), "r"(a6), "r"(a7)
2342         :);
2343     return ret;
2344 }
2345
2346 #define OPTIX_DEFINE_optixGetAttribute_BODY(which)
\
2347 unsigned int ret;
\
2348 asm("call (%0), _optix_get_attribute_" #which ", ();" : "=r"(ret) :);
\
2349     return ret;
2350
2351 static __forceinline__ __device__ unsigned int optixGetAttribute_0()
2352 {
```

```
2353     OPTIX_DEFINE_optixGetAttribute_BODY(0);
2354 }
2355
2356 static __forceinline__ __device__ unsigned int optixGetAttribute_1()
2357 {
2358     OPTIX_DEFINE_optixGetAttribute_BODY(1);
2359 }
2360
2361 static __forceinline__ __device__ unsigned int optixGetAttribute_2()
2362 {
2363     OPTIX_DEFINE_optixGetAttribute_BODY(2);
2364 }
2365
2366 static __forceinline__ __device__ unsigned int optixGetAttribute_3()
2367 {
2368     OPTIX_DEFINE_optixGetAttribute_BODY(3);
2369 }
2370
2371 static __forceinline__ __device__ unsigned int optixGetAttribute_4()
2372 {
2373     OPTIX_DEFINE_optixGetAttribute_BODY(4);
2374 }
2375
2376 static __forceinline__ __device__ unsigned int optixGetAttribute_5()
2377 {
2378     OPTIX_DEFINE_optixGetAttribute_BODY(5);
2379 }
2380
2381 static __forceinline__ __device__ unsigned int optixGetAttribute_6()
2382 {
2383     OPTIX_DEFINE_optixGetAttribute_BODY(6);
2384 }
2385
2386 static __forceinline__ __device__ unsigned int optixGetAttribute_7()
2387 {
2388     OPTIX_DEFINE_optixGetAttribute_BODY(7);
2389 }
2390
2391 #undef OPTIX_DEFINE_optixGetAttribute_BODY
2392
2393 static __forceinline__ __device__ void optixTerminateRay()
2394 {
2395     asm volatile("call _optix_terminate_ray, ();");
2396 }
2397
2398 static __forceinline__ __device__ void optixIgnoreIntersection()
2399 {
2400     asm volatile("call _optix_ignore_intersection, ();");
2401 }
2402
2403 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex()
2404 {
2405     unsigned int u0;
2406     asm("call (%0), _optix_read_primitive_idx, ();" : "=r"(u0) :);
2407     return u0;
2408 }
2409
2410 static __forceinline__ __device__ unsigned int optixGetClusterId()
2411 {
2412     unsigned int u0;
2413     asm("call (%0), _optix_get_cluster_id, ();" : "=r"(u0) :);
2414     return u0;
2415 }
2416
2417 static __forceinline__ __device__ unsigned int optixHitObjectGetClusterId()
2418 {
2419     unsigned int u0;
```

```
2420     asm("call (%0), _optix_hitobject_get_cluster_id, ();" : "=r"(u0) :);
2421     return u0;
2422 }
2423
2424 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex()
2425 {
2426     unsigned int u0;
2427     asm("call (%0), _optix_read_sbt_gas_idx, ();" : "=r"(u0) :);
2428     return u0;
2429 }
2430
2431 static __forceinline__ __device__ unsigned int optixGetInstanceId()
2432 {
2433     unsigned int u0;
2434     asm("call (%0), _optix_read_instance_id, ();" : "=r"(u0) :);
2435     return u0;
2436 }
2437
2438 static __forceinline__ __device__ unsigned int optixGetInstanceIndex()
2439 {
2440     unsigned int u0;
2441     asm("call (%0), _optix_read_instance_idx, ();" : "=r"(u0) :);
2442     return u0;
2443 }
2444
2445 static __forceinline__ __device__ unsigned int optixGetHitKind()
2446 {
2447     unsigned int u0;
2448     asm("call (%0), _optix_get_hit_kind, ();" : "=r"(u0) :);
2449     return u0;
2450 }
2451
2452 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind)
2453 {
2454     unsigned int u0;
2455     asm("call (%0), _optix_get_primitive_type_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));
2456     return (OptixPrimitiveType)u0;
2457 }
2458
2459 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind)
2460 {
2461     unsigned int u0;
2462     asm("call (%0), _optix_get_backface_from_hit_kind, (%1);" : "=r"(u0) : "r"(hitKind));
2463     return (u0 == 0x1);
2464 }
2465
2466 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind)
2467 {
2468     return !optixIsBackFaceHit(hitKind);
2469 }
2470
2471
2472 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType()
2473 {
2474     return optixGetPrimitiveType(optixGetHitKind());
2475 }
2476
2477 static __forceinline__ __device__ bool optixIsBackFaceHit()
2478 {
2479     return optixIsBackFaceHit(optixGetHitKind());
2480 }
2481
2482 static __forceinline__ __device__ bool optixIsFrontFaceHit()
2483 {
2484     return optixIsFrontFaceHit(optixGetHitKind());
2485 }
2486
```

```
2487 static __forceinline__ __device__ bool optixIsTriangleHit()
2488 {
2489     return optixIsTriangleFrontFaceHit() || optixIsTriangleBackFaceHit();
2490 }
2491
2492 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit()
2493 {
2494     return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE;
2495 }
2496
2497 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit()
2498 {
2499     return optixGetHitKind() == OPTIX_HIT_KIND_TRIANGLE_BACK_FACE;
2500 }
2501
2502
2503 static __forceinline__ __device__ float optixGetCurveParameter()
2504 {
2505     float f0;
2506     asm("call (%0), _optix_get_curve_parameter, ();" : "=f"(f0) :);
2507     return f0;
2508 }
2509
2510 static __forceinline__ __device__ float optixHitObjectGetCurveParameter()
2511 {
2512     float f0;
2513     asm("call (%0), _optix_hitobject_get_curve_parameter, ();" : "=f"(f0) :);
2514     return f0;
2515 }
2516
2517 static __forceinline__ __device__ float2 optixGetRibbonParameters()
2518 {
2519     float f0, f1;
2520     asm("call (%0, %1), _optix_get_ribbon_parameters, ();" : "=f"(f0), "=f"(f1) :);
2521     return make_float2(f0, f1);
2522 }
2523
2524 static __forceinline__ __device__ float2 optixHitObjectGetRibbonParameters()
2525 {
2526     float f0, f1;
2527     asm("call (%0, %1), _optix_hitobject_get_ribbon_parameters, ();" : "=f"(f0), "=f"(f1) :);
2528     return make_float2(f0, f1);
2529 }
2530
2531 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics()
2532 {
2533     float f0, f1;
2534     asm("call (%0, %1), _optix_get_triangle_barycentrics, ();" : "=f"(f0), "=f"(f1) :);
2535     return make_float2(f0, f1);
2536 }
2537
2538 static __forceinline__ __device__ float2 optixHitObjectGetTriangleBarycentrics()
2539 {
2540     float f0, f1;
2541     asm("call (%0, %1), _optix_hitobject_get_triangle_barycentrics, ();" : "=f"(f0), "=f"(f1) :);
2542     return make_float2(f0, f1);
2543 }
2544
2545 static __forceinline__ __device__ uint3 optixGetLaunchIndex()
2546 {
2547     unsigned int u0, u1, u2;
2548     asm("call (%0), _optix_get_launch_index_x, ();" : "=r"(u0) :);
2549     asm("call (%0), _optix_get_launch_index_y, ();" : "=r"(u1) :);
2550     asm("call (%0), _optix_get_launch_index_z, ();" : "=r"(u2) :);
2551     return make_uint3(u0, u1, u2);
2552 }
2553
```

```
2554 static __forceinline__ __device__ uint3 optixGetLaunchDimensions( )
2555 {
2556     unsigned int u0, u1, u2;
2557     asm("call (%0), _optix_get_launch_dimension_x, ();" : "=r"(u0) :);
2558     asm("call (%0), _optix_get_launch_dimension_y, ();" : "=r"(u1) :);
2559     asm("call (%0), _optix_get_launch_dimension_z, ();" : "=r"(u2) :);
2560     return make_uint3(u0, u1, u2);
2561 }
2562
2563 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer( )
2564 {
2565     unsigned long long ptr;
2566     asm("call (%0), _optix_get_sbt_data_ptr_64, ();" : "=l"(ptr) :);
2567     return (CUdeviceptr)ptr;
2568 }
2569
2570 static __forceinline__ __device__ void optixThrowException(int exceptionCode)
2571 {
2572     asm volatile(
2573         "call _optix_throw_exception_0, (%0);"
2574         : /* no return value */
2575         : "r"(exceptionCode)
2576         :);
2577 }
2578
2579 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0)
2580 {
2581     asm volatile(
2582         "call _optix_throw_exception_1, (%0, %1);"
2583         : /* no return value */
2584         : "r"(exceptionCode), "r"(exceptionDetail0)
2585         :);
2586 }
2587
2588 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1)
2589 {
2590     asm volatile(
2591         "call _optix_throw_exception_2, (%0, %1, %2);"
2592         : /* no return value */
2593         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1)
2594         :);
2595 }
2596
2597 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
2598 {
2599     asm volatile(
2600         "call _optix_throw_exception_3, (%0, %1, %2, %3);"
2601         : /* no return value */
2602         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2)
2603         :);
2604 }
2605
2606 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3)
2607 {
2608     asm volatile(
2609         "call _optix_throw_exception_4, (%0, %1, %2, %3, %4);"
2610         : /* no return value */
2611         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3)
2612         :);
2613 }
2614
```

```
2615 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int
exceptionDetail3, unsigned int exceptionDetail4)
2616 {
2617     asm volatile(
2618         "call _optix_throw_exception_5, (%0, %1, %2, %3, %4, %5);"
2619         : /* no return value */
2620         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4)
2621         :);
2622 }
2623
2624 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)
2625 {
2626     asm volatile(
2627         "call _optix_throw_exception_6, (%0, %1, %2, %3, %4, %5, %6);"
2628         : /* no return value */
2629         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5)
2630         :);
2631 }
2632
2633 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)
2634 {
2635     asm volatile(
2636         "call _optix_throw_exception_7, (%0, %1, %2, %3, %4, %5, %6, %7);"
2637         : /* no return value */
2638         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6)
2639         :);
2640 }
2641
2642 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)
2643 {
2644     asm volatile(
2645         "call _optix_throw_exception_8, (%0, %1, %2, %3, %4, %5, %6, %7, %8);"
2646         : /* no return value */
2647         : "r"(exceptionCode), "r"(exceptionDetail0), "r"(exceptionDetail1), "r"(exceptionDetail2),
"r"(exceptionDetail3), "r"(exceptionDetail4), "r"(exceptionDetail5), "r"(exceptionDetail6),
"r"(exceptionDetail7)
2648         :);
2649 }
2650
2651 static __forceinline__ __device__ int optixGetExceptionCode()
2652 {
2653     int s0;
2654     asm("call (%0), _optix_get_exception_code, ();" : "=r"(s0) :);
2655     return s0;
2656 }
2657
2658 #define OPTIX_DEFINE_optixGetExceptionDetail_BODY(which)
\
2659 unsigned int ret;
\
2660 asm("call (%0), _optix_get_exception_detail_" #which ", ();" : "=r"(ret) :);
\
2661     return ret;
2662
2663 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0()
```

```
2664 {
2665     OPTIX_DEFINE_optixGetExceptionDetail_BODY(0);
2666 }
2667
2668 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1()
2669 {
2670     OPTIX_DEFINE_optixGetExceptionDetail_BODY(1);
2671 }
2672
2673 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2()
2674 {
2675     OPTIX_DEFINE_optixGetExceptionDetail_BODY(2);
2676 }
2677
2678 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3()
2679 {
2680     OPTIX_DEFINE_optixGetExceptionDetail_BODY(3);
2681 }
2682
2683 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4()
2684 {
2685     OPTIX_DEFINE_optixGetExceptionDetail_BODY(4);
2686 }
2687
2688 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5()
2689 {
2690     OPTIX_DEFINE_optixGetExceptionDetail_BODY(5);
2691 }
2692
2693 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6()
2694 {
2695     OPTIX_DEFINE_optixGetExceptionDetail_BODY(6);
2696 }
2697
2698 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7()
2699 {
2700     OPTIX_DEFINE_optixGetExceptionDetail_BODY(7);
2701 }
2702
2703 #undef OPTIX_DEFINE_optixGetExceptionDetail_BODY
2704
2705
2706 static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable()
2707 {
2708     unsigned long long handle;
2709     asm("call (%0), _optix_get_exception_invalid_traversable, ();" : "=l"(handle) :);
2710     return (OptixTraversableHandle)handle;
2711 }
2712
2713 static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset()
2714 {
2715     int s0;
2716     asm("call (%0), _optix_get_exception_invalid_sbt_offset, ();" : "=r"(s0) :);
2717     return s0;
2718 }
2719
2720 static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay()
2721 {
2722     float rayOriginX, rayOriginY, rayOriginZ, rayDirectionX, rayDirectionY, rayDirectionZ, tmin, tmax,
rayTime;
2723     asm("call (%0, %1, %2, %3, %4, %5, %6, %7, %8), _optix_get_exception_invalid_ray, ();"
2724         : "=f"(rayOriginX), "=f"(rayOriginY), "=f"(rayOriginZ), "=f"(rayDirectionX),
"=f"(rayDirectionY),
2725           "=f"(rayDirectionZ), "=f"(tmin), "=f"(tmax), "=f"(rayTime)
2726         :);
2727     OptixInvalidRayExceptionDetails ray;
2728     ray.origin    = make_float3(rayOriginX, rayOriginY, rayOriginZ);
```

```
2729     ray.direction = make_float3(rayDirectionX, rayDirectionY, rayDirectionZ);
2730     ray.tmin      = tmin;
2731     ray.tmax      = tmax;
2732     ray.time      = rayTime;
2733     return ray;
2734 }
2735
2736 static __forceinline__ __device__ OptixParameterMismatchExceptionDetails
optixGetExceptionParameterMismatch()
2737 {
2738     unsigned int expected, actual, sbtIdx;
2739     unsigned long long calleeName;
2740     asm(
2741         "call (%0, %1, %2, %3), _optix_get_exception_parameter_mismatch, ();"
2742         : "=r"(expected), "=r"(actual), "=r"(sbtIdx), "=l"(calleeName) :);
2743     OptixParameterMismatchExceptionDetails details;
2744     details.expectedParameterCount = expected;
2745     details.passedArgumentCount = actual;
2746     details.sbtIndex = sbtIdx;
2747     details.callableName = (char*)calleeName;
2748     return details;
2749 }
2750
2751
2752 static __forceinline__ __device__ char* optixGetExceptionLineInfo()
2753 {
2754     unsigned long long ptr;
2755     asm("call (%0), _optix_get_exception_line_info, ();" : "=l"(ptr) :);
2756     return (char*)ptr;
2757 }
2758
2759 template <typename ReturnT, typename... ArgTypes>
2760 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args)
2761 {
2762     unsigned long long func;
2763     asm("call (%0), _optix_call_direct_callable,(%1);" : "=l"(func) : "r"(sbtIndex) :);
2764     using funcT = ReturnT (*)(ArgTypes...);
2765     funcT call  = (funcT)(func);
2766     return call(args...);
2767 }
2768
2769 template <typename ReturnT, typename... ArgTypes>
2770 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes... args)
2771 {
2772     unsigned long long func;
2773     asm("call (%0), _optix_call_continuation_callable,(%1);" : "=l"(func) : "r"(sbtIndex) :);
2774     using funcT = ReturnT (*)(ArgTypes...);
2775     funcT call  = (funcT)(func);
2776     return call(args...);
2777 }
2778
2779 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int
texInfo, float x, float y, unsigned int* singleMipLevel)
2780 {
2781     uint4              result;
2782     unsigned long long resultPtr         = reinterpret_cast<unsigned long long>(&result);
2783     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2784     // Cast float args to integers, because the intrinics take .b32 arguments when compiled to PTX.
2785     asm volatile(
2786         "call _optix_tex_footprint_2d_v2"
2787         ", (%0, %1, %2, %3, %4, %5);"
2788         :
2789         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2790           "l"(singleMipLevelPtr), "l"(resultPtr)
2791         :);
2792     return result;
2793 }
```

```
2794
2795 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
2796                                                                 unsigned int       texInfo,
2797                                                                 float              x,
2798                                                                 float              y,
2799                                                                 float              dPdx_x,
2800                                                                 float              dPdx_y,
2801                                                                 float              dPdy_x,
2802                                                                 float              dPdy_y,
2803                                                                 bool               coarse,
2804                                                                 unsigned int*      singleMipLevel)
2805 {
2806     uint4              result;
2807     unsigned long long resultPtr         = reinterpret_cast<unsigned long long>(&result);
2808     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2809     // Cast float args to integers, because the intrinics take .b32 arguments when compiled to PTX.
2810     asm volatile(
2811         "call _optix_tex_footprint_2d_grad_v2"
2812         ", (%0, %1, %2, %3, %4, %5, %6, %7, %8, %9, %10);"
2813         :
2814         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2815           "r"(__float_as_uint(dPdx_x)), "r"(__float_as_uint(dPdx_y)), "r"(__float_as_uint(dPdy_x)),
2816           "r"(__float_as_uint(dPdy_y)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
"l"(resultPtr)
2817         :);
2818
2819     return result;
2820 }
2821
2822 static __forceinline__ __device__ uint4
2823 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool
coarse, unsigned int* singleMipLevel)
2824 {
2825     uint4              result;
2826     unsigned long long resultPtr         = reinterpret_cast<unsigned long long>(&result);
2827     unsigned long long singleMipLevelPtr = reinterpret_cast<unsigned long long>(singleMipLevel);
2828     // Cast float args to integers, because the intrinics take .b32 arguments when compiled to PTX.
2829     asm volatile(
2830         "call _optix_tex_footprint_2d_lod_v2"
2831         ", (%0, %1, %2, %3, %4, %5, %6, %7);"
2832         :
2833         : "l"(tex), "r"(texInfo), "r"(__float_as_uint(x)), "r"(__float_as_uint(y)),
2834           "r"(__float_as_uint(level)), "r"(static_cast<unsigned int>(coarse)), "l"(singleMipLevelPtr),
"l"(resultPtr)
2835         :);
2836     return result;
2837 }
2838
2839 #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)
- template<typename HitState > static __forceinline__ __device__ void optix_impl::optixGetWorldToObjectTransformMatrix (const HitState &hs, float4 &m0, float4 &m1, float4 &m2)
- template<typename HitState > static __forceinline__ __device__ void optix_impl::optixGetObjectToWorldTransformMatrix (const HitState &hs, float4 &m0, float4 &m1, float4 &m2)
- static __forceinline__ __device__ float3 optix_impl::optixTransformPoint (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &p)
- static __forceinline__ __device__ float3 optix_impl::optixTransformVector (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &v)
- static __forceinline__ __device__ float3 optix_impl::optixTransformNormal (const float4 &m0, const float4 &m1, const float4 &m2, const float3 &n)

8.3.1 Detailed Description

OptiX public API.

Author

NVIDIA Corporation

OptiX public API Reference - Device side implementation for transformation helper functions.

8.4 optix_device_impl_transformations.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
4 *
5 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
6 * property and proprietary rights in and to this material, related
7 * documentation and any modifications thereto. Any use, reproduction,
8 * disclosure or distribution of this material and related documentation
```

```
9 * without an express license agreement from NVIDIA CORPORATION or
10 * its affiliates is strictly prohibited.
11 */
20 #if !defined(__OPTIX_INCLUDE_INTERNAL_HEADERS__)
21 #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.")
22 #endif
23
24 #ifndef OPTIX_OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H
25 #define OPTIX_OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H
26
27 namespace optix_impl {
28
29 static __forceinline__ __device__ float4 optixAddFloat4(const float4& a, const float4& b)
30 {
31     return make_float4(a.x + b.x, a.y + b.y, a.z + b.z, a.w + b.w);
32 }
33
34 static __forceinline__ __device__ float4 optixMulFloat4(const float4& a, float b)
35 {
36     return make_float4(a.x * b, a.y * b, a.z * b, a.w * b);
37 }
38
39 static __forceinline__ __device__ uint4 optixLdg(unsigned long long addr)
40 {
41     const uint4* ptr;
42     asm volatile("cvta.to.global.u64 %0, %1;" : "=l"(ptr) : "l"(addr));
43     uint4 ret;
44     asm volatile("ld.global.v4.u32 {%0,%1,%2,%3}, [%4];"
45                  : "=r"(ret.x), "=r"(ret.y), "=r"(ret.z), "=r"(ret.w)
46                  : "l"(ptr));
47     return ret;
48 }
49
50 template <class T>
51 static __forceinline__ __device__ T optixLoadReadOnlyAlign16(const T* ptr)
52 {
53     // Debug mode may keep this temporary variable
54     // If T does not enforce 16B alignment, v may not be 16B aligned and storing the loaded data from ptr
fails
55     __align__(16) T v;
56     for(int ofs                     = 0; ofs < sizeof(T); ofs += 16)
57         *(uint4*)((char*)&v + ofs) = optixLdg((unsigned long long)((char*)ptr + ofs));
58     return v;
59 }
60
61 // Multiplies the row vector vec with the 3x4 matrix with rows m0, m1, and m2
62 static __forceinline__ __device__ float4 optixMultiplyRowMatrix(const float4 vec, const float4 m0, const
float4 m1, const float4 m2)
63 {
64     float4 result;
65
66     result.x = vec.x * m0.x + vec.y * m1.x + vec.z * m2.x;
67     result.y = vec.x * m0.y + vec.y * m1.y + vec.z * m2.y;
68     result.z = vec.x * m0.z + vec.y * m1.z + vec.z * m2.z;
69     result.w = vec.x * m0.w + vec.y * m1.w + vec.z * m2.w + vec.w;
70
71     return result;
72 }
73
74 // Converts the SRT transformation srt into a 3x4 matrix with rows m0, m1, and m2
75 static __forceinline__ __device__ void optixGetMatrixFromSrt(float4& m0, float4& m1, float4& m2, const
OptixSRTData& srt)
76 {
77     // assumed to be normalized
78     const float4 q = {srt.qx, srt.qy, srt.qz, srt.qw};
79
```

```
80      const float sqw = q.w * q.w;
81      const float sqx = q.x * q.x;
82      const float sqy = q.y * q.y;
83      const float sqz = q.z * q.z;
84
85      const float xy = q.x * q.y;
86      const float zw = q.z * q.w;
87      const float xz = q.x * q.z;
88      const float yw = q.y * q.w;
89      const float yz = q.y * q.z;
90      const float xw = q.x * q.w;
91
92      m0.x = (sqx - sqy - sqz + sqw);
93      m0.y = 2.0f * (xy - zw);
94      m0.z = 2.0f * (xz + yw);
95
96      m1.x = 2.0f * (xy + zw);
97      m1.y = (-sqx + sqy - sqz + sqw);
98      m1.z = 2.0f * (yz - xw);
99
100      m2.x = 2.0f * (xz - yw);
101      m2.y = 2.0f * (yz + xw);
102      m2.z = (-sqx - sqy + sqz + sqw);
103
104      m0.w = m0.x * srt.pvx + m0.y * srt.pvy + m0.z * srt.pvz + srt.tx;
105      m1.w = m1.x * srt.pvx + m1.y * srt.pvy + m1.z * srt.pvz + srt.ty;
106      m2.w = m2.x * srt.pvx + m2.y * srt.pvy + m2.z * srt.pvz + srt.tz;
107
108      m0.z = m0.x * srt.b + m0.y * srt.c + m0.z * srt.sz;
109      m1.z = m1.x * srt.b + m1.y * srt.c + m1.z * srt.sz;
110      m2.z = m2.x * srt.b + m2.y * srt.c + m2.z * srt.sz;
111
112      m0.y = m0.x * srt.a + m0.y * srt.sy;
113      m1.y = m1.x * srt.a + m1.y * srt.sy;
114      m2.y = m2.x * srt.a + m2.y * srt.sy;
115
116      m0.x = m0.x * srt.sx;
117      m1.x = m1.x * srt.sx;
118      m2.x = m2.x * srt.sx;
119  }
120
121  // Inverts a 3x4 matrix in place
122  static __forceinline__ __device__ void optixInvertMatrix(float4& m0, float4& m1, float4& m2)
123  {
124      const float det3 =
125          m0.x * (m1.y * m2.z - m1.z * m2.y) - m0.y * (m1.x * m2.z - m1.z * m2.x) + m0.z * (m1.x * m2.y -
m1.y * m2.x);
126
127      const float inv_det3 = 1.0f / det3;
128
129      float inv3[3][3];
130      inv3[0][0] = inv_det3 * (m1.y * m2.z - m2.y * m1.z);
131      inv3[0][1] = inv_det3 * (m0.z * m2.y - m2.z * m0.y);
132      inv3[0][2] = inv_det3 * (m0.y * m1.z - m1.y * m0.z);
133
134      inv3[1][0] = inv_det3 * (m1.z * m2.x - m2.z * m1.x);
135      inv3[1][1] = inv_det3 * (m0.x * m2.z - m2.x * m0.z);
136      inv3[1][2] = inv_det3 * (m0.z * m1.x - m1.z * m0.x);
137
138      inv3[2][0] = inv_det3 * (m1.x * m2.y - m2.x * m1.y);
139      inv3[2][1] = inv_det3 * (m0.y * m2.x - m2.y * m0.x);
140      inv3[2][2] = inv_det3 * (m0.x * m1.y - m1.x * m0.y);
141
142      const float b[3] = {m0.w, m1.w, m2.w};
143
144      m0.x = inv3[0][0];
145      m0.y = inv3[0][1];
```

```
146     m0.z = inv3[0][2];
147     m0.w = -inv3[0][0] * b[0] - inv3[0][1] * b[1] - inv3[0][2] * b[2];
148
149     m1.x = inv3[1][0];
150     m1.y = inv3[1][1];
151     m1.z = inv3[1][2];
152     m1.w = -inv3[1][0] * b[0] - inv3[1][1] * b[1] - inv3[1][2] * b[2];
153
154     m2.x = inv3[2][0];
155     m2.y = inv3[2][1];
156     m2.z = inv3[2][2];
157     m2.w = -inv3[2][0] * b[0] - inv3[2][1] * b[1] - inv3[2][2] * b[2];
158 }
159
160 static __forceinline__ __device__ void optixLoadInterpolatedMatrixKey(float4& m0, float4& m1, float4&
m2, const float4* matrix, const float t1)
161 {
162     m0 = optixLoadReadOnlyAlign16(&matrix[0]);
163     m1 = optixLoadReadOnlyAlign16(&matrix[1]);
164     m2 = optixLoadReadOnlyAlign16(&matrix[2]);
165
166     // The conditional prevents concurrent loads leading to spills
167     if(t1 > 0.0f)
168     {
169         const float t0 = 1.0f - t1;
170         m0 = optixAddFloat4(optixMulFloat4(m0, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[3]),
t1));
171         m1 = optixAddFloat4(optixMulFloat4(m1, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[4]),
t1));
172         m2 = optixAddFloat4(optixMulFloat4(m2, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&matrix[5]),
t1));
173     }
174 }
175
176 static __forceinline__ __device__ void optixLoadInterpolatedSrtKey(float4&       srt0,
177                                                                     float4&       srt1,
178                                                                     float4&       srt2,
179                                                                     float4&       srt3,
180                                                                     const float4* srt,
181                                                                     const float   t1)
182 {
183     srt0 = optixLoadReadOnlyAlign16(&srt[0]);
184     srt1 = optixLoadReadOnlyAlign16(&srt[1]);
185     srt2 = optixLoadReadOnlyAlign16(&srt[2]);
186     srt3 = optixLoadReadOnlyAlign16(&srt[3]);
187
188     // The conditional prevents concurrent loads leading to spills
189     if(t1 > 0.0f)
190     {
191         const float t0 = 1.0f - t1;
192         srt0 = optixAddFloat4(optixMulFloat4(srt0, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[4]),
t1));
193         srt1 = optixAddFloat4(optixMulFloat4(srt1, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[5]),
t1));
194         srt2 = optixAddFloat4(optixMulFloat4(srt2, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[6]),
t1));
195         srt3 = optixAddFloat4(optixMulFloat4(srt3, t0), optixMulFloat4(optixLoadReadOnlyAlign16(&srt[7]),
t1));
196
197         float inv_length = 1.f / sqrt(srt2.y * srt2.y + srt2.z * srt2.z + srt2.w * srt2.w + srt3.x *
srt3.x);
198         srt2.y *= inv_length;
199         srt2.z *= inv_length;
200         srt2.w *= inv_length;
201         srt3.x *= inv_length;
202     }
203 }
```

```
204
205 static __forceinline__ __device__ void optixResolveMotionKey(float& localt, int& key, const
OptixMotionOptions& options, const float globalt)
206 {
207     const float timeBegin    = options.timeBegin;
208     const float timeEnd      = options.timeEnd;
209     const float numIntervals = (float)(options.numKeys - 1);
210
211     // No need to check the motion flags. If data originates from a valid transform list handle, then
globalt is in
212     // range, or vanish flags are not set.
213
214     // should be NaN or in [0,numIntervals]
215     float time = max(0.f, min(numIntervals, numIntervals * __fdividef(globalt - timeBegin, timeEnd -
timeBegin)));
216
217     // catch NaN (for example when timeBegin=timeEnd)
218     if(time != time)
219         time = 0.f;
220
221     const float fltKey = fminf(floorf(time), numIntervals - 1);
222
223     localt = time - fltKey;
224     key    = (int)fltKey;
225 }
226
227 // Returns the interpolated transformation matrix for a particular matrix motion transformation and point
in time.
228 static __forceinline__ __device__ void optixGetInterpolatedTransformation(float4&
trf0,
229                                                                            float4&                      trf1,
230                                                                            float4&                      trf2,
231                                                                               const OptixMatrixMotionTransform*
transformData,
232                                                                            const float                  time)
233 {
234     // Compute key and intra key time
235     float keyTime;
236     int   key;
237     optixResolveMotionKey(keyTime, key, optixLoadReadOnlyAlign16(transformData).motionOptions, time);
238
239     // Get pointer to left key
240     const float4* transform = (const float4*)(&transformData->transform[key][0]);
241
242     // Load and interpolate matrix keys
243     optixLoadInterpolatedMatrixKey(trf0, trf1, trf2, transform, keyTime);
244 }
245
246 // Returns the interpolated transformation matrix for a particular SRT motion transformation and point in
time.
247 static __forceinline__ __device__ void optixGetInterpolatedTransformation(float4&
trf0,
248                                                                              float4&                      trf1,
249                                                                              float4&                      trf2,
250                                                                                  const OptixSRTMotionTransform*
transformData,
251                                                                              const float                  time)
252 {
253     // Compute key and intra key time
254     float keyTime;
255     int   key;
256     optixResolveMotionKey(keyTime, key, optixLoadReadOnlyAlign16(transformData).motionOptions, time);
257
258     // Get pointer to left key
259     const float4* dataPtr = reinterpret_cast<const float4*>(&transformData->srtData[key]);
260
261     // Load and interpolated SRT keys
```

```
262     float4 data[4];
263     optixLoadInterpolatedSrtKey(data[0], data[1], data[2], data[3], dataPtr, keyTime);
264
265     OptixSRTData srt = {data[0].x, data[0].y, data[0].z, data[0].w, data[1].x, data[1].y, data[1].z,
data[1].w,
266                         data[2].x, data[2].y, data[2].z, data[2].w, data[3].x, data[3].y, data[3].z,
data[3].w};
267
268     // Convert SRT into a matrix
269     optixGetMatrixFromSrt(trf0, trf1, trf2, srt);
270 }
271
272 // Returns the interpolated transformation matrix for a particular traversable handle and point in time.
273 static __forceinline__ __device__ void optixGetInterpolatedTransformationFromHandle(float4&
trf0,
274                                                                                     float4&
trf1,
275                                                                                     float4&
trf2,
276                                                                                     const
OptixTraversableHandle handle,
277                                                                                     const float
time,
278                                                                                   const bool objectToWorld)
279 {
280     const OptixTransformType type = optixGetTransformTypeFromHandle(handle);
281
282     if(type == OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM || type ==
OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM)
283     {
284         if(type == OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM)
285         {
286             const OptixMatrixMotionTransform* transformData =
optixGetMatrixMotionTransformFromHandle(handle);
287             optixGetInterpolatedTransformation(trf0, trf1, trf2, transformData, time);
288         }
289         else
290         {
291             const OptixSRTMotionTransform* transformData = optixGetSRTMotionTransformFromHandle(handle);
292             optixGetInterpolatedTransformation(trf0, trf1, trf2, transformData, time);
293         }
294
295         if(!objectToWorld)
296             optixInvertMatrix(trf0, trf1, trf2);
297     }
298     else if(type == OPTIX_TRANSFORM_TYPE_INSTANCE || type == OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM)
299     {
300         const float4* transform;
301
302         if(type == OPTIX_TRANSFORM_TYPE_INSTANCE)
303         {
304             transform = (objectToWorld) ? optixGetInstanceTransformFromHandle(handle) :
305                                           optixGetInstanceInverseTransformFromHandle(handle);
306         }
307         else
308         {
309             const OptixStaticTransform* traversable = optixGetStaticTransformFromHandle(handle);
310             transform = (const float4*)((objectToWorld) ? traversable->transform :
traversable->invTransform);
311         }
312
313         trf0 = optixLoadReadOnlyAlign16(&transform[0]);
314         trf1 = optixLoadReadOnlyAlign16(&transform[1]);
315         trf2 = optixLoadReadOnlyAlign16(&transform[2]);
316     }
317     else
318     {
```

```
319         trf0 = {1.0f, 0.0f, 0.0f, 0.0f};
320         trf1 = {0.0f, 1.0f, 0.0f, 0.0f};
321         trf2 = {0.0f, 0.0f, 1.0f, 0.0f};
322     }
323 }
324
325 // Returns the world-to-object transformation matrix resulting from the transform stack and ray time of
the given hit object.
326 template<typename HitState>
327 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(const HitState& hs, float4&
m0, float4& m1, float4& m2)
328 {
329     const unsigned int size = hs.getTransformListSize();
330     const float        time = hs.getRayTime();
331
332 #pragma unroll 1
333     for(unsigned int i = 0; i < size; ++i)
334     {
335         OptixTraversableHandle handle = hs.getTransformListHandle(i);
336
337         float4 trf0, trf1, trf2;
338         optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
false);
339
340         if(i == 0)
341         {
342             m0 = trf0;
343             m1 = trf1;
344             m2 = trf2;
345         }
346         else
347         {
348             // m := trf * m
349             float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
350             m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
351             m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
352             m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
353         }
354     }
355 }
356
357 // Returns the object-to-world transformation matrix resulting from the transform stack and ray time of
the given hit object.
358 template<typename HitState>
359 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(const HitState& hs, float4&
m0, float4& m1, float4& m2)
360 {
361     const int   size = hs.getTransformListSize();
362     const float time = hs.getRayTime();
363
364 #pragma unroll 1
365     for(int i = size - 1; i >= 0; --i)
366     {
367         OptixTraversableHandle handle = hs.getTransformListHandle(i);
368
369         float4 trf0, trf1, trf2;
370         optixGetInterpolatedTransformationFromHandle(trf0, trf1, trf2, handle, time, /*objectToWorld*/
true);
371
372         if(i == size - 1)
373         {
374             m0 = trf0;
375             m1 = trf1;
376             m2 = trf2;
377         }
378         else
379         {
```

```
380             // m := trf * m
381             float4 tmp0 = m0, tmp1 = m1, tmp2 = m2;
382             m0 = optixMultiplyRowMatrix(trf0, tmp0, tmp1, tmp2);
383             m1 = optixMultiplyRowMatrix(trf1, tmp0, tmp1, tmp2);
384             m2 = optixMultiplyRowMatrix(trf2, tmp0, tmp1, tmp2);
385         }
386     }
387 }
388
389 // Multiplies the 3x4 matrix with rows m0, m1, m2 with the point p.
390 static __forceinline__ __device__ float3 optixTransformPoint(const float4& m0, const float4& m1, const
float4& m2, const float3& p)
391 {
392     float3 result;
393     result.x = m0.x * p.x + m0.y * p.y + m0.z * p.z + m0.w;
394     result.y = m1.x * p.x + m1.y * p.y + m1.z * p.z + m1.w;
395     result.z = m2.x * p.x + m2.y * p.y + m2.z * p.z + m2.w;
396     return result;
397 }
398
399 // Multiplies the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the vector v.
400 static __forceinline__ __device__ float3 optixTransformVector(const float4& m0, const float4& m1, const
float4& m2, const float3& v)
401 {
402     float3 result;
403     result.x = m0.x * v.x + m0.y * v.y + m0.z * v.z;
404     result.y = m1.x * v.x + m1.y * v.y + m1.z * v.z;
405     result.z = m2.x * v.x + m2.y * v.y + m2.z * v.z;
406     return result;
407 }
408
409 // Multiplies the transpose of the 3x3 linear submatrix of the 3x4 matrix with rows m0, m1, m2 with the
normal n.
410 // Note that the given matrix is supposed to be the inverse of the actual transformation matrix.
411 static __forceinline__ __device__ float3 optixTransformNormal(const float4& m0, const float4& m1, const
float4& m2, const float3& n)
412 {
413     float3 result;
414     result.x = m0.x * n.x + m1.x * n.y + m2.x * n.z;
415     result.y = m0.y * n.x + m1.y * n.y + m2.y * n.z;
416     result.z = m0.z * n.x + m1.z * n.y + m2.z * n.z;
417     return result;
418 }
419
420 }  // namespace optix_impl
421
422 #endif // OPTIX_OPTIX_DEVICE_IMPL_TRANSFORMATIONS_H
```

8.5 optix_micromap_impl.h File Reference

Namespaces

- namespace optix_impl

Macros

- #define OPTIX_MICROMAP_FUNC
- #define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline
- #define OPTIX_MICROMAP_FLOAT2_SUB(a, b) { a.x - b.x, a.y - b.y }

Functions

- OPTIX_MICROMAP_INLINE_FUNC float optix_impl::__uint_as_float (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::extractEvenBits (unsigned int x)

- OPTIX_MICROMAP_INLINE_FUNC unsigned int optix_impl::prefixEor (unsigned int x)
- OPTIX_MICROMAP_INLINE_FUNC void optix_impl::index2dbary (unsigned int index, unsigned int &u, unsigned int &v, unsigned int &w)
- OPTIX_MICROMAP_INLINE_FUNC void optix_impl::micro2bary (unsigned int index, unsigned int subdivisionLevel, float2 &bary0, float2 &bary1, float2 &bary2)
- OPTIX_MICROMAP_INLINE_FUNC float2 optix_impl::base2micro (const float2 &baseBarycentrics, const float2 microVertexBaseBarycentrics[3])

8.5.1 Detailed Description

OptiX micromap helper functions.

Author

NVIDIA Corporation

8.5.2 Macro Definition Documentation

8.5.2.1 OPTIX_MICROMAP_FUNC

```
#define OPTIX_MICROMAP_FUNC
```

8.6 optix_micromap_impl.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2022 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
16 * contributors may be used to endorse or promote products derived from
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
31
38 #ifndef OPTIX_OPTIX_MICROMAP_IMPL_H
39 #define OPTIX_OPTIX_MICROMAP_IMPL_H
40
41 #ifndef OPTIX_MICROMAP_FUNC
42 #ifdef __CUDACC__
```

```
43 #define OPTIX_MICROMAP_FUNC __device__
44 #else
45 #define OPTIX_MICROMAP_FUNC
46 #endif
47 #endif
48
49 namespace optix_impl {
50
55 #define OPTIX_MICROMAP_INLINE_FUNC OPTIX_MICROMAP_FUNC inline
56
57 #ifdef __CUDACC__
58 // the device implementation of __uint_as_float is declared in cuda_runtime.h
59 #else
60 // the host implementation of __uint_as_float
61 OPTIX_MICROMAP_INLINE_FUNC float __uint_as_float(unsigned int x)
62 {
63     union { float f; unsigned int i; } var;
64     var.i = x;
65     return var.f;
66 }
67 #endif
68
69 // Extract even bits
70 OPTIX_MICROMAP_INLINE_FUNC unsigned int extractEvenBits(unsigned int x)
71 {
72     x &= 0x55555555;
73     x = (x | (x » 1)) & 0x33333333;
74     x = (x | (x » 2)) & 0x0f0f0f0f;
75     x = (x | (x » 4)) & 0x00ff00ff;
76     x = (x | (x » 8)) & 0x0000ffff;
77     return x;
78 }
79
80
81 // Calculate exclusive prefix or (log(n) XOR's and SHF's)
82 OPTIX_MICROMAP_INLINE_FUNC unsigned int prefixEor(unsigned int x)
83 {
84     x ^= x » 1;
85     x ^= x » 2;
86     x ^= x » 4;
87     x ^= x » 8;
88     return x;
89 }
90
91 // Convert distance along the curve to discrete barycentrics
92 OPTIX_MICROMAP_INLINE_FUNC void index2dbary(unsigned int index, unsigned int& u, unsigned int& v, unsigned
int& w)
93 {
94     unsigned int b0 = extractEvenBits(index);
95     unsigned int b1 = extractEvenBits(index » 1);
96
97     unsigned int fx = prefixEor(b0);
98     unsigned int fy = prefixEor(b0 & ~b1);
99
100     unsigned int t = fy ^ b1;
101
102     u = (fx & ~t) | (b0 & ~t) | (~b0 & ~fx & t);
103     v = fy ^ b0;
104     w = (~fx & ~t) | (b0 & ~t) | (~b0 & fx & t);
105 }
106
107 // Compute barycentrics of a sub or micro triangle wrt a base triangle.  The order of the returned
108 // bary0, bary1, bary2 matters and allows for using this function for sub triangles and the
109 // conversion from sub triangle to base triangle barycentric space
110 OPTIX_MICROMAP_INLINE_FUNC void micro2bary(unsigned int index, unsigned int subdivisionLevel, float2&
bary0, float2& bary1, float2& bary2)
111 {
```

```
112     if(subdivisionLevel == 0)
113     {
114         bary0 = { 0, 0 };
115         bary1 = { 1, 0 };
116         bary2 = { 0, 1 };
117         return;
118     }
119
120     unsigned int iu, iv, iw;
121     index2dbary(index, iu, iv, iw);
122
123     // we need to only look at "level" bits
124     iu = iu & ((1 « subdivisionLevel) - 1);
125     iv = iv & ((1 « subdivisionLevel) - 1);
126     iw = iw & ((1 « subdivisionLevel) - 1);
127
128     int yFlipped = (iu & 1) ^ (iv & 1) ^ (iw & 1) ^ 1;
129
130     int xFlipped = ((0x8888888888888888ull ^ 0xf000f000f000f000ull ^ 0xffff000000000000ull) » index) & 1;
131     xFlipped    ^= ((0x8888888888888888ull ^ 0xf000f000f000f000ull ^ 0xffff000000000000ull) » (index »
6)) & 1;
132
133     const float levelScale = __uint_as_float((127u - subdivisionLevel) « 23);
134
135     // scale the barycentic coordinate to the global space/scale
136     float du = 1.f * levelScale;
137     float dv = 1.f * levelScale;
138
139     // scale the barycentic coordinate to the global space/scale
140     float u = (float)iu * levelScale;
141     float v = (float)iv * levelScale;
142
143     //     c        d
144     //      x-----x
145     //     / \   /
146     //    /   \ /
147     //   x-----x
148     //  a        b
149     //
150     // !xFlipped && !yFlipped: abc
151     // !xFlipped &&  yFlipped: cdb
152     //  xFlipped && !yFlipped: bac
153     //  xFlipped &&  yFlipped: dcb
154
155     bary0 = { u + xFlipped * du    , v + yFlipped * dv };
156     bary1 = { u + (1-xFlipped) * du, v + yFlipped * dv };
157     bary2 = { u + yFlipped * du    , v + (1-yFlipped) * dv };
158 }
159
160 // avoid any conflicts due to multiple definitions
161 #define OPTIX_MICROMAP_FLOAT2_SUB(a,b) { a.x - b.x, a.y - b.y }
162
163 // Compute barycentrics for micro triangle from base barycentrics
164 OPTIX_MICROMAP_INLINE_FUNC float2 base2micro(const float2& baseBarycentrics, const float2
microVertexBaseBarycentrics[3])
165 {
166     float2 baryV0P  = OPTIX_MICROMAP_FLOAT2_SUB(baseBarycentrics, microVertexBaseBarycentrics[0]);
167     float2 baryV0V1 = OPTIX_MICROMAP_FLOAT2_SUB(microVertexBaseBarycentrics[1],
microVertexBaseBarycentrics[0]);
168     float2 baryV0V2 = OPTIX_MICROMAP_FLOAT2_SUB(microVertexBaseBarycentrics[2],
microVertexBaseBarycentrics[0]);
169
170     float  rdetA = 1.f / (baryV0V1.x * baryV0V2.y - baryV0V1.y * baryV0V2.x);
171     float4 A     = { baryV0V2.y, -baryV0V2.x, -baryV0V1.y, baryV0V1.x };
172
173     float2 localUV;
174     localUV.x = rdetA * (baryV0P.x * A.x + baryV0P.y * A.y);
```

```
175     localUV.y = rdetA * (baryV0P.x * A.z + baryV0P.y * A.w);
176
177     return localUV;
178 }
179 #undef OPTIX_MICROMAP_FLOAT2_SUB
180   // end group optix_utilities
182
183 }  // namespace optix_impl
184
185 #endif  // OPTIX_OPTIX_MICROMAP_IMPL_H
```

8.7 optix.h File Reference

Macros

- #define OPTIX_VERSION 90000

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 90000
```

The OptiX version.

- major = OPTIX_VERSION/10000
- minor = (OPTIX_VERSION%10000)/100
- micro = OPTIX_VERSION%100

8.8 optix.h

Go to the documentation of this file.

```
1
2 /*
3 * SPDX-FileCopyrightText: Copyright (c) 2009 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
4 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
5 *
6 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
7 * property and proprietary rights in and to this material, related
8 * documentation and any modifications thereto. Any use, reproduction,
9 * disclosure or distribution of this material and related documentation
10 * without an express license agreement from NVIDIA CORPORATION or
11 * its affiliates is strictly prohibited.
12 */
19
20 #ifndef OPTIX_OPTIX_H
21 #define OPTIX_OPTIX_H
22
28 #define OPTIX_VERSION 90000
29
```

```
30
31 #ifdef __CUDACC__
32 #include "optix_device.h"
33 #else
34 #include "optix_host.h"
35 #endif
36
37
38 #endif  // OPTIX_OPTIX_H
```

8.9 optix_denoiser_tiling.h File Reference

Classes

- struct OptixUtilDenoiserImageTile

Functions

- OptixResult optixUtilGetPixelStride (const OptixImage2D &image, unsigned int &pixelStrideInBytes)
- OptixResult optixUtilDenoiserSplitImage (const OptixImage2D &input, const OptixImage2D &output, unsigned int overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight, std::vector< OptixUtilDenoiserImageTile > &tiles)
- OptixResult optixUtilDenoiserInvokeTiled (OptixDenoiser denoiser, CUstream stream, const OptixDenoiserParams *params, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, const OptixDenoiserGuideLayer *guideLayer, const OptixDenoiserLayer *layers, unsigned int numLayers, CUdeviceptr scratch, size_t scratchSizeInBytes, unsigned int overlapWindowSizeInPixels, unsigned int tileWidth, unsigned int tileHeight)

8.9.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.10 optix_denoiser_tiling.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
16 * contributors may be used to endorse or promote products derived from
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
```

```
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
34
35 #ifndef OPTIX_DENOISER_TILING_H
36 #define OPTIX_DENOISER_TILING_H
37
38 #include <optix.h>
39
40 #include <algorithm>
41 #include <vector>
42
43 #ifdef __cplusplus
44 extern "C" {
45 #endif
46
55 struct OptixUtilDenoiserImageTile
56 {
57     // input tile image
58     OptixImage2D input;
59
60     // output tile image
61     OptixImage2D output;
62
63     // overlap offsets, parameters for #optixUtilDenoiserInvoke
64     unsigned int inputOffsetX;
65     unsigned int inputOffsetY;
66 };
67
75 inline OptixResult optixUtilGetPixelStride(const OptixImage2D& image, unsigned int& pixelStrideInBytes)
76 {
77     pixelStrideInBytes = image.pixelStrideInBytes;
78     if(pixelStrideInBytes == 0)
79     {
80         switch(image.format)
81         {
82             case OPTIX_PIXEL_FORMAT_HALF1:
83                 pixelStrideInBytes = 1 * sizeof(short);
84                 break;
85             case OPTIX_PIXEL_FORMAT_HALF2:
86                 pixelStrideInBytes = 2 * sizeof(short);
87                 break;
88             case OPTIX_PIXEL_FORMAT_HALF3:
89                 pixelStrideInBytes = 3 * sizeof(short);
90                 break;
91             case OPTIX_PIXEL_FORMAT_HALF4:
92                 pixelStrideInBytes = 4 * sizeof(short);
93                 break;
94             case OPTIX_PIXEL_FORMAT_FLOAT1:
95                 pixelStrideInBytes = 1 * sizeof(float);
96                 break;
97             case OPTIX_PIXEL_FORMAT_FLOAT2:
98                 pixelStrideInBytes = 2 * sizeof(float);
99                 break;
100             case OPTIX_PIXEL_FORMAT_FLOAT3:
101                 pixelStrideInBytes = 3 * sizeof(float);
102                 break;
103             case OPTIX_PIXEL_FORMAT_FLOAT4:
104                 pixelStrideInBytes = 4 * sizeof(float);
105                 break;
106             case OPTIX_PIXEL_FORMAT_UCHAR3:
```

```
107                 pixelStrideInBytes = 3 * sizeof(char);
108                 break;
109             case OPTIX_PIXEL_FORMAT_UCHAR4:
110                 pixelStrideInBytes = 4 * sizeof(char);
111                 break;
112             case OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER:
113                 return OPTIX_ERROR_INVALID_VALUE;
114                 break;
115         }
116     }
117     return OPTIX_SUCCESS;
118 }
119
129 inline OptixResult optixUtilDenoiserSplitImage(
130                                                 const OptixImage2D&                    input,
131                                                 const OptixImage2D&                    output,
132                                           unsigned int                         overlapWindowSizeInPixels,
133                                                 unsigned int                           tileWidth,
134                                                 unsigned int                           tileHeight,
135                                                 std::vector<OptixUtilDenoiserImageTile>&    tiles)
136 {
137     if(tileWidth == 0 || tileHeight == 0)
138         return OPTIX_ERROR_INVALID_VALUE;
139
140     unsigned int inPixelStride, outPixelStride;
141     if(const OptixResult res = optixUtilGetPixelStride(input, inPixelStride))
142         return res;
143     if(const OptixResult res = optixUtilGetPixelStride(output, outPixelStride))
144         return res;
145
146     int inp_w = std::min(tileWidth + 2 * overlapWindowSizeInPixels, input.width);
147     int inp_h = std::min(tileHeight + 2 * overlapWindowSizeInPixels, input.height);
148     int inp_y = 0, copied_y = 0;
149
150     int upscaleX = output.width / input.width;
151     int upscaleY = output.height / input.height;
152
153     do
154     {
155         int inputOffsetY = inp_y == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_h -
((int)input.height - inp_y));
156         int copy_y       = inp_y == 0 ? std::min(input.height, tileHeight + overlapWindowSizeInPixels) :
157                                         std::min(tileHeight, input.height - copied_y);
158
159         int inp_x = 0, copied_x = 0;
160         do
161         {
162             int inputOffsetX = inp_x == 0 ? 0 : std::max((int)overlapWindowSizeInPixels, inp_w -
((int)input.width - inp_x));
163             int copy_x = inp_x == 0 ? std::min(input.width, tileWidth + overlapWindowSizeInPixels) :
164                                       std::min(tileWidth, input.width - copied_x);
165
166             OptixUtilDenoiserImageTile tile;
167             tile.input.data              = input.data + (size_t)(inp_y - inputOffsetY) *
input.rowStrideInBytes
168                                            + (size_t)(inp_x - inputOffsetX) * inPixelStride;
169             tile.input.width             = inp_w;
170             tile.input.height            = inp_h;
171             tile.input.rowStrideInBytes   = input.rowStrideInBytes;
172             tile.input.pixelStrideInBytes = input.pixelStrideInBytes;
173             tile.input.format            = input.format;
174
175             tile.output.data              = output.data + (size_t)(upscaleY * inp_y) *
output.rowStrideInBytes
176                                             + (size_t)(upscaleX * inp_x) * outPixelStride;
177             tile.output.width             = upscaleX * copy_x;
178             tile.output.height            = upscaleY * copy_y;
```

```
179             tile.output.rowStrideInBytes   = output.rowStrideInBytes;
180             tile.output.pixelStrideInBytes = output.pixelStrideInBytes;
181             tile.output.format             = output.format;
182 
183             tile.inputOffsetX = inputOffsetX;
184             tile.inputOffsetY = inputOffsetY;
185 
186             tiles.push_back(tile);
187 
188             inp_x += inp_x == 0 ? tileWidth + overlapWindowSizeInPixels : tileWidth;
189             copied_x += copy_x;
190         } while(inp_x < static_cast<int>(input.width));
191 
192         inp_y += inp_y == 0 ? tileHeight + overlapWindowSizeInPixels : tileHeight;
193         copied_y += copy_y;
194     } while(inp_y < static_cast<int>(input.height));
195 
196     return OPTIX_SUCCESS;
197 }
198 
202 
209 
225 inline OptixResult optixUtilDenoiserInvokeTiled(
226                                                 OptixDenoiser                   denoiser,
227                                                 CUstream                        stream,
228                                                 const OptixDenoiserParams*      params,
229                                                 CUdeviceptr                     denoiserState,
230                                                 size_t                         denoiserStateSizeInBytes,
231                                                 const OptixDenoiserGuideLayer*  guideLayer,
232                                                 const OptixDenoiserLayer*       layers,
233                                                 unsigned int                    numLayers,
234                                                 CUdeviceptr                     scratch,
235                                                 size_t                          scratchSizeInBytes,
236                                                unsigned int                   overlapWindowSizeInPixels,
237                                                 unsigned int                    tileWidth,
238                                                 unsigned int                    tileHeight)
239 {
240     if(!guideLayer || !layers)
241         return OPTIX_ERROR_INVALID_VALUE;
242 
243     const unsigned int upscale = numLayers > 0 && layers[0].previousOutput.width == 2 *
layers[0].input.width ? 2 : 1;
244 
245     std::vector<std::vector<OptixUtilDenoiserImageTile» tiles(numLayers);
246     std::vector<std::vector<OptixUtilDenoiserImageTile» prevTiles(numLayers);
247     for(unsigned int l = 0; l < numLayers; l++)
248     {
249         if(const OptixResult res = optixUtilDenoiserSplitImage(layers[l].input, layers[l].output,
250                                                                overlapWindowSizeInPixels,
251                                                                tileWidth, tileHeight, tiles[l]))
252             return res;
253 
254         if(layers[l].previousOutput.data)
255         {
256             OptixImage2D dummyOutput = layers[l].previousOutput;
257            if(const OptixResult res = optixUtilDenoiserSplitImage(layers[l].previousOutput, dummyOutput,
258                                                                upscale * overlapWindowSizeInPixels,
259                                                               upscale * tileWidth, upscale * tileHeight,
prevTiles[l]))
260                 return res;
261         }
262     }
263 
264     std::vector<OptixUtilDenoiserImageTile> albedoTiles;
265     if(guideLayer->albedo.data)
266     {
267         OptixImage2D dummyOutput = guideLayer->albedo;
```

```
268         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->albedo, dummyOutput,
269                                                                overlapWindowSizeInPixels,
270                                                                tileWidth, tileHeight, albedoTiles))
271             return res;
272     }
273
274     std::vector<OptixUtilDenoiserImageTile> normalTiles;
275     if(guideLayer->normal.data)
276     {
277         OptixImage2D dummyOutput = guideLayer->normal;
278         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->normal, dummyOutput,
279                                                                overlapWindowSizeInPixels,
280                                                                tileWidth, tileHeight, normalTiles))
281             return res;
282     }
283
284     std::vector<OptixUtilDenoiserImageTile> flowTiles;
285     if(guideLayer->flow.data)
286     {
287         OptixImage2D dummyOutput = guideLayer->flow;
288         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->flow, dummyOutput,
289                                                                overlapWindowSizeInPixels,
290                                                                tileWidth, tileHeight, flowTiles))
291             return res;
292     }
293
294     std::vector<OptixUtilDenoiserImageTile> flowTrustTiles;
295     if(guideLayer->flowTrustworthiness.data)
296     {
297         OptixImage2D dummyOutput = guideLayer->flowTrustworthiness;
298         if(const OptixResult res = optixUtilDenoiserSplitImage(guideLayer->flowTrustworthiness,
dummyOutput,
299                                                                overlapWindowSizeInPixels,
300                                                                tileWidth, tileHeight, flowTrustTiles))
301             return res;
302     }
303
304     std::vector<OptixUtilDenoiserImageTile> internalGuideLayerTiles;
305     if(guideLayer->previousOutputInternalGuideLayer.data && guideLayer->outputInternalGuideLayer.data)
306     {
307         if(const OptixResult res =
optixUtilDenoiserSplitImage(guideLayer->previousOutputInternalGuideLayer,
308                                                                guideLayer->outputInternalGuideLayer,
309                                                                upscale * overlapWindowSizeInPixels,
310                                                               upscale * tileWidth, upscale * tileHeight,
internalGuideLayerTiles))
311             return res;
312     }
313
314     for(size_t t = 0; t < tiles[0].size(); t++)
315     {
316         std::vector<OptixDenoiserLayer> tlayers;
317         for(unsigned int l = 0; l < numLayers; l++)
318         {
319             OptixDenoiserLayer layer = {};
320             layer.input  = (tiles[l])[t].input;
321             layer.output = (tiles[l])[t].output;
322             if(layers[l].previousOutput.data)
323                 layer.previousOutput = (prevTiles[l])[t].input;
324             layer.type = layers[l].type;
325             tlayers.push_back(layer);
326         }
327
328         OptixDenoiserGuideLayer gl = {};
329         if(guideLayer->albedo.data)
330             gl.albedo = albedoTiles[t].input;
331
```

```
332         if(guideLayer->normal.data)
333             gl.normal = normalTiles[t].input;
334 
335         if(guideLayer->flow.data)
336             gl.flow = flowTiles[t].input;
337 
338         if(guideLayer->flowTrustworthiness.data)
339             gl.flowTrustworthiness = flowTrustTiles[t].input;
340 
341         if(guideLayer->previousOutputInternalGuideLayer.data)
342             gl.previousOutputInternalGuideLayer = internalGuideLayerTiles[t].input;
343 
344         if(guideLayer->outputInternalGuideLayer.data)
345             gl.outputInternalGuideLayer = internalGuideLayerTiles[t].output;
346 
347         if(const OptixResult res =
348                 optixDenoiserInvoke(denoiser, stream, params, denoiserState, denoiserStateSizeInBytes,
349                                     &gl, &tlayers[0], numLayers,
350                                     (tiles[0])[t].inputOffsetX, (tiles[0])[t].inputOffsetY,
351                                     scratch, scratchSizeInBytes))
352             return res;
353     }
354     return OPTIX_SUCCESS;
355 }
356   // end group optix_utilities
358 
359 #ifdef __cplusplus
360 }
361 #endif
362 
363 #endif  // OPTIX_DENOISER_TILING_H
```

8.11 optix_device.h File Reference

Classes

- struct OptixIncomingHitObject
- struct OptixOutgoingHitObject
- class OptixCoopVec< T, N >

Macros

- #define __OPTIX_INCLUDE_INTERNAL_HEADERS__
- #define OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET
- #define OPTIX_INCLUDE_COOPERATIVE_VECTOR 1

Functions

- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTraverse (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixTrace (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float

rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)

- template<typename... Payload>
  static __forceinline__ __device__ void optixTraverse (OptixPayloadTypeID type, OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, OptixVisibilityMask visibilityMask, unsigned int rayFlags, unsigned int SBToffset, unsigned int SBTstride, unsigned int missSBTIndex, Payload &... payload)
- static __forceinline__ __device__ void optixReorder (unsigned int coherenceHint, unsigned int numCoherenceHintBitsFromLSB)
- static __forceinline__ __device__ void optixReorder ()
- template<typename... Payload>
  static __forceinline__ __device__ void optixInvoke (Payload &... payload)
- template<typename... Payload>
  static __forceinline__ __device__ void optixInvoke (OptixPayloadTypeID type, Payload &... payload)
- static __forceinline__ __device__ void optixMakeHitObject (OptixTraversableHandle handle, float3 rayOrigin, float3 rayDirection, float tmin, float rayTime, unsigned int rayFlags, OptixTraverseData traverseData, const OptixTraversableHandle *transforms, unsigned int numTransforms)
- static __forceinline__ __device__ void optixMakeMissHitObject (unsigned int missSBTIndex, float3 rayOrigin, float3 rayDirection, float tmin, float tmax, float rayTime, unsigned int rayFlags)
- static __forceinline__ __device__ void optixMakeNopHitObject ()
- static __forceinline__ __device__ void optixHitObjectGetTraverseData (OptixTraverseData *data)
- static __forceinline__ __device__ bool optixHitObjectIsHit ()
- static __forceinline__ __device__ bool optixHitObjectIsMiss ()
- static __forceinline__ __device__ bool optixHitObjectIsNop ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex ()
- static __forceinline__ __device__ void optixHitObjectSetSbtRecordIndex (unsigned int sbtRecordIndex)
- static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetGASTraversableHandle ()
- static __forceinline__ __device__ void optixSetPayload_0 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_1 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_2 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_3 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_4 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_5 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_6 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_7 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_8 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_9 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_10 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_11 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_12 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_13 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_14 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_15 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_16 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_17 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_18 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_19 (unsigned int p)

- static __forceinline__ __device__ void optixSetPayload_20 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_21 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_22 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_23 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_24 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_25 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_26 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_27 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_28 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_29 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_30 (unsigned int p)
- static __forceinline__ __device__ void optixSetPayload_31 (unsigned int p)
- static __forceinline__ __device__ unsigned int optixGetPayload_0 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_1 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_2 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_3 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_4 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_5 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_6 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_7 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_8 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_9 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_10 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_11 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_12 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_13 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_14 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_15 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_16 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_17 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_18 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_19 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_20 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_21 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_22 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_23 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_24 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_25 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_26 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_27 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_28 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_29 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_30 ()
- static __forceinline__ __device__ unsigned int optixGetPayload_31 ()
- static __forceinline__ __device__ void optixSetPayloadTypes (unsigned int typeMask)
- static __forceinline__ __device__ unsigned int optixUndefinedValue ()
- static __forceinline__ __device__ float3 optixGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin ()
- static __forceinline__ __device__ float3 optixGetWorldRayDirection ()
- static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection ()

- static __forceinline__ __device__ float3 optixGetObjectRayOrigin ()
- static __forceinline__ __device__ float3 optixGetObjectRayDirection ()
- static __forceinline__ __device__ float optixGetRayTmin ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmin ()
- static __forceinline__ __device__ float optixGetRayTmax ()
- static __forceinline__ __device__ float optixHitObjectGetRayTmax ()
- static __forceinline__ __device__ float optixGetRayTime ()
- static __forceinline__ __device__ float optixHitObjectGetRayTime ()
- static __forceinline__ __device__ unsigned int optixGetRayFlags ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetRayFlags ()
- static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceTraversableFromIAS (OptixTraversableHandle ias, unsigned int instIdx)
- static __forceinline__ __device__ void optixGetTriangleVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static __forceinline__ __device__ void optixGetTriangleVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float3 data[3])
- static __forceinline__ __device__ void optixGetTriangleVertexData (float3 data[3])
- static __forceinline__ __device__ void optixHitObjectGetTriangleVertexData (float3 data[3])
- static __forceinline__ __device__ void optixGetLinearCurveVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void optixGetLinearCurveVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[2])
- static __forceinline__ __device__ void optixGetLinearCurveVertexData (float4 data[2])
- static __forceinline__ __device__ void optixHitObjectGetLinearCurveVertexData (float4 data[2])
- static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetQuadraticBSplineVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData (float4 data[3])
- static __forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexData (float4 data[3])
- static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineVertexData (float4 data[3])
- static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineRocapsVertexData (float4 data[3])
- static __forceinline__ __device__ void optixGetCubicBSplineVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBSplineVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])

- static __forceinline__ __device__ void optixGetCubicBSplineVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCubicBSplineVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCubicBSplineRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCatmullRomVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCatmullRomRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCubicBezierVertexData (float4 data[4])
- static __forceinline__ __device__ void optixHitObjectGetCubicBezierRocapsVertexData (float4 data[4])
- static __forceinline__ __device__ void optixGetRibbonVertexData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetRibbonVertexDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[3])
- static __forceinline__ __device__ void optixGetRibbonVertexData (float4 data[3])
- static __forceinline__ __device__ void optixHitObjectGetRibbonVertexData (float4 data[3])
- static __forceinline__ __device__ float3 optixGetRibbonNormal (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ float3 optixGetRibbonNormalFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float2 ribbonParameters)
- static __forceinline__ __device__ float3 optixGetRibbonNormal (float2 ribbonParameters)
- static __forceinline__ __device__ float3 optixHitObjectGetRibbonNormal (float2 ribbonParameters)
- static __forceinline__ __device__ void optixGetSphereData (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ void optixGetSphereDataFromHandle (OptixTraversableHandle gas, unsigned int primIdx, unsigned int sbtGASIndex, float time, float4 data[1])
- static __forceinline__ __device__ void optixGetSphereData (float4 data[1])

- static __forceinline__ __device__ void optixHitObjectGetSphereData (float4 data[1])
- static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle ()
- static __forceinline__ __device__ float optixGetGASMotionTimeBegin (OptixTraversableHandle gas)
- static __forceinline__ __device__ float optixGetGASMotionTimeEnd (OptixTraversableHandle gas)
- static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount (OptixTraversableHandle gas)
- static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (float m[12])
- static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (float3 normal)
- static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (float3 point)
- static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (float3 vec)
- static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (float3 normal)
- static __forceinline__ __device__ void optixHitObjectGetWorldToObjectTransformMatrix (float m[12])
- static __forceinline__ __device__ void optixHitObjectGetObjectToWorldTransformMatrix (float m[12])
- static __forceinline__ __device__ float3 optixHitObjectTransformPointFromWorldToObjectSpace (float3 point)
- static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromWorldToObjectSpace (float3 vec)
- static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromWorldToObjectSpace (float3 normal)
- static __forceinline__ __device__ float3 optixHitObjectTransformPointFromObjectToWorldSpace (float3 point)
- static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromObjectToWorldSpace (float3 vec)
- static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromObjectToWorldSpace (float3 normal)
- template<typename HitState >
  static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix (const HitState &hs, float m[12])
- template<typename HitState >
  static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix (const HitState &hs, float m[12])
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace (const HitState &hs, float3 point)
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace (const HitState &hs, float3 vec)

- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace (const HitState &hs, float3 normal)
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace (const HitState &hs, float3 point)
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace (const HitState &hs, float3 vec)
- template<typename HitState >
  static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace (const HitState &hs, float3 normal)
- static __forceinline__ __device__ unsigned int optixGetTransformListSize ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle (unsigned int index)
- static __forceinline__ __device__ OptixTransformType optixGetTransformTypeFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixStaticTransform * optixGetStaticTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixSRTMotionTransform * optixGetSRTMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const OptixMatrixMotionTransform * optixGetMatrixMotionTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ OptixTraversableHandle optixGetInstanceChildFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceTransformFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ const float4 * optixGetInstanceInverseTransformFromHandle (OptixTraversableHandle handle)
- static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle (OptixTraversableHandle handle)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5)

- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6)
- static __forceinline__ __device__ bool optixReportIntersection (float hitT, unsigned int hitKind, unsigned int a0, unsigned int a1, unsigned int a2, unsigned int a3, unsigned int a4, unsigned int a5, unsigned int a6, unsigned int a7)
- static __forceinline__ __device__ unsigned int optixGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixGetAttribute_7 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6 ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7 ()
- static __forceinline__ __device__ void optixTerminateRay ()
- static __forceinline__ __device__ void optixIgnoreIntersection ()
- static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixGetClusterId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetClusterId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex ()
- static __forceinline__ __device__ unsigned int optixGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex ()
- static __forceinline__ __device__ unsigned int optixGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId ()
- static __forceinline__ __device__ unsigned int optixGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex ()
- static __forceinline__ __device__ unsigned int optixGetHitKind ()
- static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind ()
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsFrontFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ bool optixIsBackFaceHit (unsigned int hitKind)
- static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType ()
- static __forceinline__ __device__ bool optixIsFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsBackFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleHit ()
- static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit ()
- static __forceinline__ __device__ bool optixIsTriangleBackFaceHit ()
- static __forceinline__ __device__ float2 optixGetTriangleBarycentrics ()
- static __forceinline__ __device__ float2 optixHitObjectGetTriangleBarycentrics ()
- static __forceinline__ __device__ float optixGetCurveParameter ()
- static __forceinline__ __device__ float optixHitObjectGetCurveParameter ()

- static __forceinline__ __device__ float2 optixGetRibbonParameters ()
- static __forceinline__ __device__ float2 optixHitObjectGetRibbonParameters ()
- static __forceinline__ __device__ uint3 optixGetLaunchIndex ()
- static __forceinline__ __device__ uint3 optixGetLaunchDimensions ()
- static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer ()
- static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer ()
- static __forceinline__ __device__ void optixThrowException (int exceptionCode)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6)
- static __forceinline__ __device__ void optixThrowException (int exceptionCode, unsigned int exceptionDetail0, unsigned int exceptionDetail1, unsigned int exceptionDetail2, unsigned int exceptionDetail3, unsigned int exceptionDetail4, unsigned int exceptionDetail5, unsigned int exceptionDetail6, unsigned int exceptionDetail7)
- static __forceinline__ __device__ int optixGetExceptionCode ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6 ()
- static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7 ()
- static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable ()
- static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset ()
- static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay ()
- static __forceinline__ __device__ OptixParameterMismatchExceptionDetails optixGetExceptionParameterMismatch ()
- static __forceinline__ __device__ char * optixGetExceptionLineInfo ()
- template<typename ReturnT , typename... ArgTypes>
  static __forceinline__ __device__ ReturnT optixDirectCall (unsigned int sbtIndex, ArgTypes... args)

- template<typename ReturnT , typename... ArgTypes>
  static __forceinline__ __device__ ReturnT optixContinuationCall (unsigned int sbtIndex, ArgTypes... args)
- static __forceinline__ __device__ uint4 optixTexFootprint2D (unsigned long long tex, unsigned int texInfo, float x, float y, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DLod (unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool coarse, unsigned int *singleMipLevel)
- static __forceinline__ __device__ uint4 optixTexFootprint2DGrad (unsigned long long tex, unsigned int texInfo, float x, float y, float dPdx_x, float dPdx_y, float dPdy_x, float dPdy_y, bool coarse, unsigned int *singleMipLevel)
- template<typename VecTOut >
  static __forceinline__ __device__ VecTOut optixCoopVecLoad (CUdeviceptr ptr)
- template<typename VecTOut , typename T >
  static __forceinline__ __device__ VecTOut optixCoopVecLoad (T *ptr)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecExp2 (const VecT &vec)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecLog2 (const VecT &vec)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecTanh (const VecT &vec)
- template<typename VecTOut , typename VecTIn >
  static __forceinline__ __device__ VecTOut optixCoopVecCvt (const VecTIn &vec)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecMin (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecMin (const VecT &vecA, typename VecT ::value_type B)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecMax (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecMax (const VecT &vecA, typename VecT ::value_type B)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecMul (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecAdd (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecSub (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecStep (const VecT &vecA, const VecT &vecB)
- template<typename VecT >
  static __forceinline__ __device__ VecT optixCoopVecFFMA (const VecT &vecA, const VecT &vecB, const VecT &vecC)
- template<typename VecTOut , typename VecTIn , OptixCoopVecElemType inputInterpretation, OptixCoopVecMatrixLayout matrixLayout, bool transpose, unsigned int N, unsigned int K, OptixCoopVecElemType matrixElementType, OptixCoopVecElemType biasElementType>
  static __forceinline__ __device__ VecTOut optixCoopVecMatMul (const VecTIn &inputVector, CUdeviceptr matrix, unsigned matrixOffsetInBytes, CUdeviceptr bias, unsigned biasOffsetInBytes, unsigned rowColumnStrideInBytes=0)
- template<typename VecTIn >
  static __forceinline__ __device__ void optixCoopVecReduceSumAccumulate (const VecTIn &inputVector, CUdeviceptr outputVector, unsigned offsetInBytes)

- template<typename VecTA , typename VecTB , OptixCoopVecMatrixLayout matrixLayout = OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL>
  static __forceinline__ __device__ void optixCoopVecOuterProductAccumulate (const VecTA &vecA, const VecTB &vecB, CUdeviceptr outputMatrix, unsigned offsetInBytes, unsigned rowColumnStrideInBytes=0)
- template<unsigned int N, unsigned int K, OptixCoopVecElemType elementType, OptixCoopVecMatrixLayout layout = OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL, unsigned int rowColumnStrideInBytes = 0>
  static __forceinline__ __device__ unsigned int optixCoopVecGetMatrixSize ()

8.11.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

OptiX public API Reference - Device API declarations

8.11.2 Macro Definition Documentation

8.11.2.1 __OPTIX_INCLUDE_INTERNAL_HEADERS__

```
#define __OPTIX_INCLUDE_INTERNAL_HEADERS__
```

8.11.2.2 OPTIX_INCLUDE_COOPERATIVE_VECTOR

```
#define OPTIX_INCLUDE_COOPERATIVE_VECTOR 1
```

8.11.2.3 OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET

```
#define OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET
```

8.12 optix_device.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2010 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
4 *
5 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
6 * property and proprietary rights in and to this material, related
7 * documentation and any modifications thereto. Any use, reproduction,
8 * disclosure or distribution of this material and related documentation
9 * without an express license agreement from NVIDIA CORPORATION or
10 * its affiliates is strictly prohibited.
11 */
17
18 #ifndef OPTIX_OPTIX_DEVICE_H
19 #define OPTIX_OPTIX_DEVICE_H
20
21 #if defined(__cplusplus) && (__cplusplus < 201103L) && !defined(_WIN32)
22 #error Device code for OptiX requires at least C++11. Consider adding "--std c++11" to the nvcc
command-line.
23 #endif
24
25 #include "optix_types.h"
26
29
```

```
51 template <typename... Payload>
52 static __forceinline__ __device__ void optixTrace(OptixTraversableHandle handle,
53                                                    float3                 rayOrigin,
54                                                    float3                 rayDirection,
55                                                    float                  tmin,
56                                                    float                  tmax,
57                                                    float                  rayTime,
58                                                    OptixVisibilityMask    visibilityMask,
59                                                    unsigned int           rayFlags,
60                                                    unsigned int           SBToffset,
61                                                    unsigned int           SBTstride,
62                                                    unsigned int           missSBTIndex,
63                                                    Payload&...              payload);
64
84 template <typename... Payload>
85 static __forceinline__ __device__ void optixTraverse(OptixTraversableHandle handle,
86                                                       float3                 rayOrigin,
87                                                       float3                 rayDirection,
88                                                       float                  tmin,
89                                                       float                  tmax,
90                                                       float                  rayTime,
91                                                       OptixVisibilityMask    visibilityMask,
92                                                       unsigned int           rayFlags,
93                                                       unsigned int           SBToffset,
94                                                       unsigned int           SBTstride,
95                                                       unsigned int           missSBTIndex,
96                                                       Payload&... payload);
97
115 template <typename... Payload>
116 static __forceinline__ __device__ void optixTrace(OptixPayloadTypeID     type,
117                                                    OptixTraversableHandle handle,
118                                                    float3                 rayOrigin,
119                                                    float3                 rayDirection,
120                                                    float                  tmin,
121                                                    float                  tmax,
122                                                    float                  rayTime,
123                                                    OptixVisibilityMask    visibilityMask,
124                                                    unsigned int           rayFlags,
125                                                    unsigned int           SBToffset,
126                                                    unsigned int           SBTstride,
127                                                    unsigned int           missSBTIndex,
128                                                    Payload&...              payload);
129
150 template <typename... Payload>
151 static __forceinline__ __device__ void optixTraverse(OptixPayloadTypeID     type,
152                                                       OptixTraversableHandle handle,
153                                                       float3                 rayOrigin,
154                                                       float3                 rayDirection,
155                                                       float                  tmin,
156                                                       float                  tmax,
157                                                       float                  rayTime,
158                                                       OptixVisibilityMask    visibilityMask,
159                                                       unsigned int           rayFlags,
160                                                       unsigned int           SBToffset,
161                                                       unsigned int           SBTstride,
162                                                       unsigned int           missSBTIndex,
163                                                       Payload&... payload);
164
175 static __forceinline__ __device__ void optixReorder(unsigned int coherenceHint, unsigned int
numCoherenceHintBitsFromLSB);
176
180 static __forceinline__ __device__ void optixReorder();
181
190 template <typename... Payload>
191 static __forceinline__ __device__ void optixInvoke(Payload&... payload);
192
202 template <typename... Payload>
```

```
203 static __forceinline__ __device__ void optixInvoke(OptixPayloadTypeID type, Payload&... payload);
204
220 static __forceinline__ __device__ void optixMakeHitObject(OptixTraversableHandle        handle,
221                                                            float3                         rayOrigin,
222                                                            float3                         rayDirection,
223                                                            float                          tmin,
224                                                            float                          rayTime,
225                                                            unsigned int                   rayFlags,
226                                                            OptixTraverseData              traverseData,
227                                                            const OptixTraversableHandle* transforms,
228                                                            unsigned int                   numTransforms);
229
243 static __forceinline__ __device__ void optixMakeMissHitObject(unsigned int missSBTIndex,
244                                                                float3       rayOrigin,
245                                                                float3       rayDirection,
246                                                                float        tmin,
247                                                                float        tmax,
248                                                                float        rayTime,
249                                                                unsigned int rayFlags);
250
258 static __forceinline__ __device__ void optixMakeNopHitObject();
259
266 static __forceinline__ __device__ void optixHitObjectGetTraverseData(OptixTraverseData* data);
267
271 static __forceinline__ __device__ bool optixHitObjectIsHit();
272
276 static __forceinline__ __device__ bool optixHitObjectIsMiss();
277
283 static __forceinline__ __device__ bool optixHitObjectIsNop();
284
291 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtRecordIndex();
292
296 static __forceinline__ __device__ void optixHitObjectSetSbtRecordIndex(unsigned int sbtRecordIndex);
297
303 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetGASTraversableHandle();
304
310 static __forceinline__ __device__ void optixSetPayload_0(unsigned int p);
311 static __forceinline__ __device__ void optixSetPayload_1(unsigned int p);
312 static __forceinline__ __device__ void optixSetPayload_2(unsigned int p);
313 static __forceinline__ __device__ void optixSetPayload_3(unsigned int p);
314 static __forceinline__ __device__ void optixSetPayload_4(unsigned int p);
315 static __forceinline__ __device__ void optixSetPayload_5(unsigned int p);
316 static __forceinline__ __device__ void optixSetPayload_6(unsigned int p);
317 static __forceinline__ __device__ void optixSetPayload_7(unsigned int p);
318 static __forceinline__ __device__ void optixSetPayload_8(unsigned int p);
319 static __forceinline__ __device__ void optixSetPayload_9(unsigned int p);
320 static __forceinline__ __device__ void optixSetPayload_10(unsigned int p);
321 static __forceinline__ __device__ void optixSetPayload_11(unsigned int p);
322 static __forceinline__ __device__ void optixSetPayload_12(unsigned int p);
323 static __forceinline__ __device__ void optixSetPayload_13(unsigned int p);
324 static __forceinline__ __device__ void optixSetPayload_14(unsigned int p);
325 static __forceinline__ __device__ void optixSetPayload_15(unsigned int p);
326 static __forceinline__ __device__ void optixSetPayload_16(unsigned int p);
327 static __forceinline__ __device__ void optixSetPayload_17(unsigned int p);
328 static __forceinline__ __device__ void optixSetPayload_18(unsigned int p);
329 static __forceinline__ __device__ void optixSetPayload_19(unsigned int p);
330 static __forceinline__ __device__ void optixSetPayload_20(unsigned int p);
331 static __forceinline__ __device__ void optixSetPayload_21(unsigned int p);
332 static __forceinline__ __device__ void optixSetPayload_22(unsigned int p);
333 static __forceinline__ __device__ void optixSetPayload_23(unsigned int p);
334 static __forceinline__ __device__ void optixSetPayload_24(unsigned int p);
335 static __forceinline__ __device__ void optixSetPayload_25(unsigned int p);
336 static __forceinline__ __device__ void optixSetPayload_26(unsigned int p);
337 static __forceinline__ __device__ void optixSetPayload_27(unsigned int p);
338 static __forceinline__ __device__ void optixSetPayload_28(unsigned int p);
339 static __forceinline__ __device__ void optixSetPayload_29(unsigned int p);
340 static __forceinline__ __device__ void optixSetPayload_30(unsigned int p);
```

```
341 static __forceinline__ __device__ void optixSetPayload_31(unsigned int p);
342
348 static __forceinline__ __device__ unsigned int optixGetPayload_0();
349 static __forceinline__ __device__ unsigned int optixGetPayload_1();
350 static __forceinline__ __device__ unsigned int optixGetPayload_2();
351 static __forceinline__ __device__ unsigned int optixGetPayload_3();
352 static __forceinline__ __device__ unsigned int optixGetPayload_4();
353 static __forceinline__ __device__ unsigned int optixGetPayload_5();
354 static __forceinline__ __device__ unsigned int optixGetPayload_6();
355 static __forceinline__ __device__ unsigned int optixGetPayload_7();
356 static __forceinline__ __device__ unsigned int optixGetPayload_8();
357 static __forceinline__ __device__ unsigned int optixGetPayload_9();
358 static __forceinline__ __device__ unsigned int optixGetPayload_10();
359 static __forceinline__ __device__ unsigned int optixGetPayload_11();
360 static __forceinline__ __device__ unsigned int optixGetPayload_12();
361 static __forceinline__ __device__ unsigned int optixGetPayload_13();
362 static __forceinline__ __device__ unsigned int optixGetPayload_14();
363 static __forceinline__ __device__ unsigned int optixGetPayload_15();
364 static __forceinline__ __device__ unsigned int optixGetPayload_16();
365 static __forceinline__ __device__ unsigned int optixGetPayload_17();
366 static __forceinline__ __device__ unsigned int optixGetPayload_18();
367 static __forceinline__ __device__ unsigned int optixGetPayload_19();
368 static __forceinline__ __device__ unsigned int optixGetPayload_20();
369 static __forceinline__ __device__ unsigned int optixGetPayload_21();
370 static __forceinline__ __device__ unsigned int optixGetPayload_22();
371 static __forceinline__ __device__ unsigned int optixGetPayload_23();
372 static __forceinline__ __device__ unsigned int optixGetPayload_24();
373 static __forceinline__ __device__ unsigned int optixGetPayload_25();
374 static __forceinline__ __device__ unsigned int optixGetPayload_26();
375 static __forceinline__ __device__ unsigned int optixGetPayload_27();
376 static __forceinline__ __device__ unsigned int optixGetPayload_28();
377 static __forceinline__ __device__ unsigned int optixGetPayload_29();
378 static __forceinline__ __device__ unsigned int optixGetPayload_30();
379 static __forceinline__ __device__ unsigned int optixGetPayload_31();
380
389 static __forceinline__ __device__ void optixSetPayloadTypes(unsigned int typeMask);
390
394 static __forceinline__ __device__ unsigned int optixUndefinedValue();
395
402 static __forceinline__ __device__ float3 optixGetWorldRayOrigin();
403
409 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayOrigin();
410
417 static __forceinline__ __device__ float3 optixGetWorldRayDirection();
418
424 static __forceinline__ __device__ float3 optixHitObjectGetWorldRayDirection();
425
429 static __forceinline__ __device__ float3 optixGetObjectRayOrigin();
430
434 static __forceinline__ __device__ float3 optixGetObjectRayDirection();
435
439 static __forceinline__ __device__ float optixGetRayTmin();
440
446 static __forceinline__ __device__ float optixHitObjectGetRayTmin();
447
456 static __forceinline__ __device__ float optixGetRayTmax();
457
466 static __forceinline__ __device__ float optixHitObjectGetRayTmax();
467
473 static __forceinline__ __device__ float optixGetRayTime();
474
480 static __forceinline__ __device__ float optixHitObjectGetRayTime();
481
485 static __forceinline__ __device__ unsigned int optixGetRayFlags();
486
490 static __forceinline__ __device__ unsigned int optixHitObjectGetRayFlags();
491
```

```
495 static __forceinline__ __device__ unsigned int optixGetRayVisibilityMask();
496
503 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceTraversableFromIAS(OptixTraversableHandle ias, unsigned int instIdx);
504
517 static __forceinline__ __device__ void optixGetTriangleVertexData(OptixTraversableHandle gas,
518                                                                    unsigned int           primIdx,
519                                                                    unsigned int           sbtGASIndex,
520                                                                    float                  time,
521                                                                    float3                 data[3]);
522
536 static __forceinline__ __device__ void optixGetTriangleVertexDataFromHandle(OptixTraversableHandle gas,
537                                                                              unsigned int           primIdx,
538                                                                            unsigned int           sbtGASIndex,
539                                                                                  float                  time,
540                                                                               float3                 data[3]);
541
550 static __forceinline__ __device__ void optixGetTriangleVertexData(float3 data[3]);
551
558 static __forceinline__ __device__ void optixHitObjectGetTriangleVertexData(float3 data[3]);
559
560
576 static __forceinline__ __device__ void optixGetLinearCurveVertexData(OptixTraversableHandle gas,
577                                                                      unsigned int           primIdx,
578                                                                     unsigned int           sbtGASIndex,
579                                                                      float                  time,
580                                                                      float4                 data[2]);
581
597 static __forceinline__ __device__ void optixGetLinearCurveVertexDataFromHandle(OptixTraversableHandle
gas,
598                                                                                unsigned int           primIdx,
599                                                                              unsigned int           sbtGASIndex,
600                                                                                  float                  time,
601                                                                               float4                 data[2]);
602
611 static __forceinline__ __device__ void optixGetLinearCurveVertexData(float4 data[2]);
612
619 static __forceinline__ __device__ void optixHitObjectGetLinearCurveVertexData(float4 data[2]);
620
633 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(OptixTraversableHandle gas,
634                                                                            unsigned int           primIdx,
635                                                                          unsigned int           sbtGASIndex,
636                                                                               float                  time,
637                                                                            float4                 data[3]);
638
651 static __forceinline__ __device__ void
optixGetQuadraticBSplineVertexDataFromHandle(OptixTraversableHandle gas,
652                                                                                 unsigned int           primIdx,
653                                                                                           unsigned int
sbtGASIndex,
654                                                                                    float                  time,
655                                                                                           float4
data[3]);
656 static __forceinline__ __device__ void
optixGetQuadraticBSplineRocapsVertexDataFromHandle(OptixTraversableHandle gas,
657                                                                                       unsigned int primIdx,
658                                                                                              unsigned int
sbtGASIndex,
659                                                                                          float        time,
660                                                                                       float4       data[3]);
661
668 static __forceinline__ __device__ void optixGetQuadraticBSplineVertexData(float4 data[3]);
669 static __forceinline__ __device__ void optixGetQuadraticBSplineRocapsVertexData(float4 data[3]);
670
679 static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineVertexData(float4 data[3]);
680 static __forceinline__ __device__ void optixHitObjectGetQuadraticBSplineRocapsVertexData(float4 data[3]);
681
```

```
697 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(OptixTraversableHandle gas,
698                                                                         unsigned int           primIdx,
699                                                                       unsigned int           sbtGASIndex,
700                                                                         float                  time,
701                                                                         float4                 data[4]);
702
715 static __forceinline__ __device__ void optixGetCubicBSplineVertexDataFromHandle(OptixTraversableHandle
gas,
716                                                                                  unsigned int          primIdx,
717                                                                                        unsigned int
sbtGASIndex,
718                                                                                     float                 time,
719                                                                                   float4                data[4]);
720 static __forceinline__ __device__ void
optixGetCubicBSplineRocapsVertexDataFromHandle(OptixTraversableHandle gas,
721                                                                                                unsigned int
primIdx,
722                                                                                           unsigned int sbtGASIndex,
723                                                                                                 float        time,
724                                                                                              float4       data[4]);
725
732 static __forceinline__ __device__ void optixGetCubicBSplineVertexData(float4 data[4]);
733 static __forceinline__ __device__ void optixGetCubicBSplineRocapsVertexData(float4 data[4]);
734
744 static __forceinline__ __device__ void optixHitObjectGetCubicBSplineVertexData(float4 data[4]);
751 static __forceinline__ __device__ void optixHitObjectGetCubicBSplineRocapsVertexData(float4 data[4]);
752
768 static __forceinline__ __device__ void optixGetCatmullRomVertexData(OptixTraversableHandle gas,
769                                                                     unsigned int           primIdx,
770                                                                     unsigned int           sbtGASIndex,
771                                                                     float                  time,
772                                                                     float4                 data[4]);
773
786 static __forceinline__ __device__ void optixGetCatmullRomVertexDataFromHandle(OptixTraversableHandle gas,
787                                                                                unsigned int          primIdx,
788                                                                              unsigned int          sbtGASIndex,
789                                                                                 float                 time,
790                                                                               float4                data[4]);
791 static __forceinline__ __device__ void
optixGetCatmullRomRocapsVertexDataFromHandle(OptixTraversableHandle gas,
792                                                                                 unsigned int          primIdx,
793                                                                                           unsigned int
sbtGASIndex,
794                                                                                    float                 time,
795                                                                                           float4
data[4]);
796
803 static __forceinline__ __device__ void optixGetCatmullRomVertexData(float4 data[4]);
804 static __forceinline__ __device__ void optixGetCatmullRomRocapsVertexData(float4 data[4]);
805
815 static __forceinline__ __device__ void optixHitObjectGetCatmullRomVertexData(float4 data[4]);
816 static __forceinline__ __device__ void optixHitObjectGetCatmullRomRocapsVertexData(float4 data[4]);
817
833 static __forceinline__ __device__ void optixGetCubicBezierVertexData(OptixTraversableHandle gas,
834                                                                       unsigned int           primIdx,
835                                                                     unsigned int           sbtGASIndex,
836                                                                       float                  time,
837                                                                       float4                 data[4]);
838
851 static __forceinline__ __device__ void optixGetCubicBezierVertexDataFromHandle(OptixTraversableHandle
gas,
852                                                                                unsigned int          primIdx,
853                                                                              unsigned int          sbtGASIndex,
854                                                                                 float                 time,
855                                                                               float4                data[4]);
856 static __forceinline__ __device__ void
optixGetCubicBezierRocapsVertexDataFromHandle(OptixTraversableHandle gas,
```

```
857                                                                                 unsigned int
primIdx,
858                                                                               unsigned int sbtGASIndex,
859                                                                                     float        time,
860                                                                                  float4       data[4]);
861
868 static __forceinline__ __device__ void optixGetCubicBezierVertexData(float4 data[4]);
869 static __forceinline__ __device__ void optixGetCubicBezierRocapsVertexData(float4 data[4]);
870
880 static __forceinline__ __device__ void optixHitObjectGetCubicBezierVertexData(float4 data[4]);
881 static __forceinline__ __device__ void optixHitObjectGetCubicBezierRocapsVertexData(float4 data[4]);
882
898 static __forceinline__ __device__ void optixGetRibbonVertexData(OptixTraversableHandle gas,
899                                                                 unsigned int           primIdx,
900                                                                 unsigned int           sbtGASIndex,
901                                                                 float                  time,
902                                                                 float4                 data[3]);
903
916 static __forceinline__ __device__ void optixGetRibbonVertexDataFromHandle(OptixTraversableHandle gas,
917                                                                           unsigned int           primIdx,
918                                                                         unsigned int           sbtGASIndex,
919                                                                              float                  time,
920                                                                           float4                 data[3]);
921
928 static __forceinline__ __device__ void optixGetRibbonVertexData(float4 data[3]);
929
939 static __forceinline__ __device__ void optixHitObjectGetRibbonVertexData(float4 data[3]);
940
947 static __forceinline__ __device__ float3 optixGetRibbonNormal(OptixTraversableHandle gas,
948                                                               unsigned int           primIdx,
949                                                               unsigned int           sbtGASIndex,
950                                                               float                  time,
951                                                               float2                 ribbonParameters);
952
956 static __forceinline__ __device__ float3 optixGetRibbonNormalFromHandle(OptixTraversableHandle gas,
957                                                                         unsigned int           primIdx,
958                                                                       unsigned int           sbtGASIndex,
959                                                                          float                  time,
960                                                                    float2                 ribbonParameters);
961
965 static __forceinline__ __device__ float3 optixGetRibbonNormal(float2 ribbonParameters);
966
970 static __forceinline__ __device__ float3 optixHitObjectGetRibbonNormal(float2 ribbonParameters);
971
987 static __forceinline__ __device__ void optixGetSphereData(OptixTraversableHandle gas,
988                                                           unsigned int           primIdx,
989                                                           unsigned int           sbtGASIndex,
990                                                           float                  time,
991                                                           float4                 data[1]);
992
1008 static __forceinline__ __device__ void optixGetSphereDataFromHandle(OptixTraversableHandle gas,
1009                                                                     unsigned int           primIdx,
1010                                                                   unsigned int           sbtGASIndex,
1011                                                                    float                  time,
1012                                                                    float4                 data[1]);
1013
1022 static __forceinline__ __device__ void optixGetSphereData(float4 data[1]);
1023
1030 static __forceinline__ __device__ void optixHitObjectGetSphereData(float4 data[1]);
1031
1036 static __forceinline__ __device__ OptixTraversableHandle optixGetGASTraversableHandle();
1037
1041 static __forceinline__ __device__ float optixGetGASMotionTimeBegin(OptixTraversableHandle gas);
1042
1046 static __forceinline__ __device__ float optixGetGASMotionTimeEnd(OptixTraversableHandle gas);
1047
1051 static __forceinline__ __device__ unsigned int optixGetGASMotionStepCount(OptixTraversableHandle gas);
```

```
1052
1059 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(float m[12]);
1060
1067 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(float m[12]);
1068
1075 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(float3 point);
1076
1083 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(float3 vec);
1084
1091 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(float3 normal);
1092
1099 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(float3 point);
1100
1107 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(float3 vec);
1108
1115 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(float3 normal);
1116
1123 static __forceinline__ __device__ void optixHitObjectGetWorldToObjectTransformMatrix(float m[12]);
1124
1131 static __forceinline__ __device__ void optixHitObjectGetObjectToWorldTransformMatrix(float m[12]);
1132
1139 static __forceinline__ __device__ float3 optixHitObjectTransformPointFromWorldToObjectSpace(float3
point);
1140
1147 static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromWorldToObjectSpace(float3
vec);
1148
1155 static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromWorldToObjectSpace(float3
normal);
1156
1163 static __forceinline__ __device__ float3 optixHitObjectTransformPointFromObjectToWorldSpace(float3
point);
1164
1171 static __forceinline__ __device__ float3 optixHitObjectTransformVectorFromObjectToWorldSpace(float3
vec);
1172
1179 static __forceinline__ __device__ float3 optixHitObjectTransformNormalFromObjectToWorldSpace(float3
normal);
1180
1195 template <typename HitState>
1196 static __forceinline__ __device__ void optixGetWorldToObjectTransformMatrix(const HitState& hs, float
m[12]);
1197
1204 template <typename HitState>
1205 static __forceinline__ __device__ void optixGetObjectToWorldTransformMatrix(const HitState& hs, float
m[12]);
1206
1213 template <typename HitState>
1214 static __forceinline__ __device__ float3 optixTransformPointFromWorldToObjectSpace(const HitState& hs,
float3 point);
1215
1222 template <typename HitState>
1223 static __forceinline__ __device__ float3 optixTransformVectorFromWorldToObjectSpace(const HitState& hs,
float3 vec);
1224
1231 template <typename HitState>
1232 static __forceinline__ __device__ float3 optixTransformNormalFromWorldToObjectSpace(const HitState& hs,
float3 normal);
1233
1240 template <typename HitState>
1241 static __forceinline__ __device__ float3 optixTransformPointFromObjectToWorldSpace(const HitState& hs,
float3 point);
1242
1249 template <typename HitState>
1250 static __forceinline__ __device__ float3 optixTransformVectorFromObjectToWorldSpace(const HitState& hs,
float3 vec);
1251
```

```
1258 template <typename HitState>
1259 static __forceinline__ __device__ float3 optixTransformNormalFromObjectToWorldSpace(const HitState& hs,
float3 normal);
1260
1264 static __forceinline__ __device__ unsigned int optixGetTransformListSize();
1265
1274 static __forceinline__ __device__ unsigned int optixHitObjectGetTransformListSize();
1275
1279 static __forceinline__ __device__ OptixTraversableHandle optixGetTransformListHandle(unsigned int
index);
1280
1289 static __forceinline__ __device__ OptixTraversableHandle optixHitObjectGetTransformListHandle(unsigned
int index);
1290
1291 struct OptixIncomingHitObject
1292 {
1293     __forceinline__ __device__ float        getRayTime()const { return optixGetRayTime(); }
1294     __forceinline__ __device__ unsigned int getTransformListSize()const { return
optixGetTransformListSize(); }
1295     __forceinline__ __device__ OptixTraversableHandle getTransformListHandle(unsigned int index)const
1296 {
1297         return optixGetTransformListHandle(index);
1298     }
1299 };
1300
1301 struct OptixOutgoingHitObject
1302 {
1303     __forceinline__ __device__ float        getRayTime()const { return optixHitObjectGetRayTime(); }
1304     __forceinline__ __device__ unsigned int getTransformListSize()const
1305 {
1306         return optixHitObjectGetTransformListSize();
1307     }
1308     __forceinline__ __device__ OptixTraversableHandle getTransformListHandle(unsigned int index)const
1309 {
1310         return optixHitObjectGetTransformListHandle(index);
1311     }
1312 };
1313
1317 static __forceinline__ __device__ OptixTransformType
optixGetTransformTypeFromHandle(OptixTraversableHandle handle);
1318
1324 static __forceinline__ __device__ const OptixStaticTransform*
optixGetStaticTransformFromHandle(OptixTraversableHandle handle);
1325
1331 static __forceinline__ __device__ const OptixSRTMotionTransform*
optixGetSRTMotionTransformFromHandle(OptixTraversableHandle handle);
1332
1338 static __forceinline__ __device__ const OptixMatrixMotionTransform*
optixGetMatrixMotionTransformFromHandle(OptixTraversableHandle handle);
1339
1345 static __forceinline__ __device__ unsigned int optixGetInstanceIdFromHandle(OptixTraversableHandle
handle);
1346
1352 static __forceinline__ __device__ OptixTraversableHandle
optixGetInstanceChildFromHandle(OptixTraversableHandle handle);
1353
1359 static __forceinline__ __device__ const float4*
optixGetInstanceTransformFromHandle(OptixTraversableHandle handle);
1360
1366 static __forceinline__ __device__ const float4*
optixGetInstanceInverseTransformFromHandle(OptixTraversableHandle handle);
1367
1373 static __device__ __forceinline__ CUdeviceptr optixGetGASPointerFromHandle(OptixTraversableHandle
handle);
1397 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind);
1398
1404 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
```

```
unsigned int a0);
1405
1411 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1);
1412
1418 static __forceinline__ __device__ bool optixReportIntersection(float hitT, unsigned int hitKind,
unsigned int a0, unsigned int a1, unsigned int a2);
1419
1425 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1426                                                                unsigned int hitKind,
1427                                                                unsigned int a0,
1428                                                                unsigned int a1,
1429                                                                unsigned int a2,
1430                                                                unsigned int a3);
1431
1437 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1438                                                                unsigned int hitKind,
1439                                                                unsigned int a0,
1440                                                                unsigned int a1,
1441                                                                unsigned int a2,
1442                                                                unsigned int a3,
1443                                                                unsigned int a4);
1444
1450 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1451                                                                unsigned int hitKind,
1452                                                                unsigned int a0,
1453                                                                unsigned int a1,
1454                                                                unsigned int a2,
1455                                                                unsigned int a3,
1456                                                                unsigned int a4,
1457                                                                unsigned int a5);
1458
1464 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1465                                                                unsigned int hitKind,
1466                                                                unsigned int a0,
1467                                                                unsigned int a1,
1468                                                                unsigned int a2,
1469                                                                unsigned int a3,
1470                                                                unsigned int a4,
1471                                                                unsigned int a5,
1472                                                                unsigned int a6);
1473
1479 static __forceinline__ __device__ bool optixReportIntersection(float        hitT,
1480                                                                unsigned int hitKind,
1481                                                                unsigned int a0,
1482                                                                unsigned int a1,
1483                                                                unsigned int a2,
1484                                                                unsigned int a3,
1485                                                                unsigned int a4,
1486                                                                unsigned int a5,
1487                                                                unsigned int a6,
1488                                                                unsigned int a7);
1489
1494 static __forceinline__ __device__ unsigned int optixGetAttribute_0();
1495 static __forceinline__ __device__ unsigned int optixGetAttribute_1();
1496 static __forceinline__ __device__ unsigned int optixGetAttribute_2();
1497 static __forceinline__ __device__ unsigned int optixGetAttribute_3();
1498 static __forceinline__ __device__ unsigned int optixGetAttribute_4();
1499 static __forceinline__ __device__ unsigned int optixGetAttribute_5();
1500 static __forceinline__ __device__ unsigned int optixGetAttribute_6();
1501 static __forceinline__ __device__ unsigned int optixGetAttribute_7();
1502
1503
1511 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_0();
1512 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_1();
1513 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_2();
1514 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_3();
```

```
1515 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_4();
1516 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_5();
1517 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_6();
1518 static __forceinline__ __device__ unsigned int optixHitObjectGetAttribute_7();
1519
1523 static __forceinline__ __device__ void optixTerminateRay();
1524
1529 static __forceinline__ __device__ void optixIgnoreIntersection();
1530
1531
1547 static __forceinline__ __device__ unsigned int optixGetPrimitiveIndex();
1548
1549
1555 static __forceinline__ __device__ unsigned int optixGetClusterId();
1556
1563 static __forceinline__ __device__ unsigned int optixHitObjectGetClusterId();
1564
1565
1573 static __forceinline__ __device__ unsigned int optixHitObjectGetPrimitiveIndex();
1574
1585 static __forceinline__ __device__ unsigned int optixGetSbtGASIndex();
1586
1595 static __forceinline__ __device__ unsigned int optixHitObjectGetSbtGASIndex();
1596
1597
1610 static __forceinline__ __device__ unsigned int optixGetInstanceId();
1611
1620 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceId();
1621
1631 static __forceinline__ __device__ unsigned int optixGetInstanceIndex();
1632
1641 static __forceinline__ __device__ unsigned int optixHitObjectGetInstanceIndex();
1642
1650 static __forceinline__ __device__ unsigned int optixGetHitKind();
1651
1659 static __forceinline__ __device__ unsigned int optixHitObjectGetHitKind();
1660
1664 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType(unsigned int hitKind);
1665
1669 static __forceinline__ __device__ bool optixIsFrontFaceHit(unsigned int hitKind);
1670
1674 static __forceinline__ __device__ bool optixIsBackFaceHit(unsigned int hitKind);
1675
1679 static __forceinline__ __device__ OptixPrimitiveType optixGetPrimitiveType();
1680
1684 static __forceinline__ __device__ bool optixIsFrontFaceHit();
1685
1689 static __forceinline__ __device__ bool optixIsBackFaceHit();
1690
1694 static __forceinline__ __device__ bool optixIsTriangleHit();
1695
1699 static __forceinline__ __device__ bool optixIsTriangleFrontFaceHit();
1700
1704 static __forceinline__ __device__ bool optixIsTriangleBackFaceHit();
1705
1706
1713 static __forceinline__ __device__ float2 optixGetTriangleBarycentrics();
1714
1722 static __forceinline__ __device__ float2 optixHitObjectGetTriangleBarycentrics();
1723
1728 static __forceinline__ __device__ float optixGetCurveParameter();
1729
1738 static __forceinline__ __device__ float optixHitObjectGetCurveParameter();
1739
1745 static __forceinline__ __device__ float2 optixGetRibbonParameters();
1746
1755 static __forceinline__ __device__ float2 optixHitObjectGetRibbonParameters();
```

```
1756
1763 static __forceinline__ __device__ uint3 optixGetLaunchIndex();
1764
1769 static __forceinline__ __device__ uint3 optixGetLaunchDimensions();
1770
1778 static __forceinline__ __device__ CUdeviceptr optixGetSbtDataPointer();
1779
1786 static __forceinline__ __device__ CUdeviceptr optixHitObjectGetSbtDataPointer();
1787
1802 static __forceinline__ __device__ void optixThrowException(int exceptionCode);
1803
1809 static __forceinline__ __device__ void optixThrowException(int exceptionCode, unsigned int
exceptionDetail0);
1810
1816 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1817                                                            unsigned int exceptionDetail0,
1818                                                            unsigned int exceptionDetail1);
1819
1825 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1826                                                            unsigned int exceptionDetail0,
1827                                                            unsigned int exceptionDetail1,
1828                                                            unsigned int exceptionDetail2);
1829
1835 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1836                                                            unsigned int exceptionDetail0,
1837                                                            unsigned int exceptionDetail1,
1838                                                            unsigned int exceptionDetail2,
1839                                                            unsigned int exceptionDetail3);
1840
1846 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1847                                                            unsigned int exceptionDetail0,
1848                                                            unsigned int exceptionDetail1,
1849                                                            unsigned int exceptionDetail2,
1850                                                            unsigned int exceptionDetail3,
1851                                                            unsigned int exceptionDetail4);
1852
1858 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1859                                                            unsigned int exceptionDetail0,
1860                                                            unsigned int exceptionDetail1,
1861                                                            unsigned int exceptionDetail2,
1862                                                            unsigned int exceptionDetail3,
1863                                                            unsigned int exceptionDetail4,
1864                                                            unsigned int exceptionDetail5);
1865
1872 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1873                                                            unsigned int exceptionDetail0,
1874                                                            unsigned int exceptionDetail1,
1875                                                            unsigned int exceptionDetail2,
1876                                                            unsigned int exceptionDetail3,
1877                                                            unsigned int exceptionDetail4,
1878                                                            unsigned int exceptionDetail5,
1879                                                            unsigned int exceptionDetail6);
1880
1886 static __forceinline__ __device__ void optixThrowException(int exceptionCode,
1887                                                            unsigned int exceptionDetail0,
1888                                                            unsigned int exceptionDetail1,
1889                                                            unsigned int exceptionDetail2,
1890                                                            unsigned int exceptionDetail3,
1891                                                            unsigned int exceptionDetail4,
1892                                                            unsigned int exceptionDetail5,
1893                                                            unsigned int exceptionDetail6,
1894                                                            unsigned int exceptionDetail7);
1895
1899 static __forceinline__ __device__ int optixGetExceptionCode();
1900
1907 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_0();
1908
```

```
1914 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_1();
1915
1921 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_2();
1922
1928 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_3();
1929
1935 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_4();
1936
1942 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_5();
1943
1949 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_6();
1950
1956 static __forceinline__ __device__ unsigned int optixGetExceptionDetail_7();
1957
1958
1964 static __forceinline__ __device__ OptixTraversableHandle optixGetExceptionInvalidTraversable();
1965
1973 static __forceinline__ __device__ int optixGetExceptionInvalidSbtOffset();
1974
1983 static __forceinline__ __device__ OptixInvalidRayExceptionDetails optixGetExceptionInvalidRay();
1984
1996 static __forceinline__ __device__ OptixParameterMismatchExceptionDetails
optixGetExceptionParameterMismatch();
1997
1998
2011 static __forceinline__ __device__ char* optixGetExceptionLineInfo();
2012
2036 template <typename ReturnT, typename... ArgTypes>
2037 static __forceinline__ __device__ ReturnT optixDirectCall(unsigned int sbtIndex, ArgTypes... args);
2038
2039
2062 template <typename ReturnT, typename... ArgTypes>
2063 static __forceinline__ __device__ ReturnT optixContinuationCall(unsigned int sbtIndex, ArgTypes...
args);
2064
2065
2130 static __forceinline__ __device__ uint4 optixTexFootprint2D(unsigned long long tex, unsigned int
texInfo, float x, float y, unsigned int* singleMipLevel);
2131
2143 static __forceinline__ __device__ uint4
2144 optixTexFootprint2DLod(unsigned long long tex, unsigned int texInfo, float x, float y, float level, bool
coarse, unsigned int* singleMipLevel);
2145
2160 static __forceinline__ __device__ uint4 optixTexFootprint2DGrad(unsigned long long tex,
2161                                                                 unsigned int       texInfo,
2162                                                                 float              x,
2163                                                                 float              y,
2164                                                                 float              dPdx_x,
2165                                                                 float              dPdx_y,
2166                                                                 float              dPdy_x,
2167                                                                 float              dPdy_y,
2168                                                                 bool               coarse,
2169                                                                 unsigned int*      singleMipLevel);
2170   // end group optix_device_api
2172
2173 #define __OPTIX_INCLUDE_INTERNAL_HEADERS__
2174
2175 #include "internal/optix_device_impl.h"
2176
2177
2178 // If you manually define OPTIX_INCLUDE_COOPERATIVE_VECTOR to override the default behavior, you must
2179 // set it to 0 or 1 and not simply define it with no value (which will default it have a value of 0).
2180 #ifndef OPTIX_INCLUDE_COOPERATIVE_VECTOR
2181 #  define OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET
2182 #  ifdef __CUDACC_RTC__ // NVRTC and cooperative vectors are currently unsupported
2183 #    define OPTIX_INCLUDE_COOPERATIVE_VECTOR 0
2184 #  else
```

```
2185 #    define OPTIX_INCLUDE_COOPERATIVE_VECTOR 1
2186 #  endif
2187 #endif
2188
2189 #if OPTIX_INCLUDE_COOPERATIVE_VECTOR
2195
2200 template <typename VecTOut>
2201 static __forceinline__ __device__ VecTOut optixCoopVecLoad(CUdeviceptr ptr);
2206 template <typename VecTOut, typename T>
2207 static __forceinline__ __device__ VecTOut optixCoopVecLoad(T* ptr);
2208
2209
2215 template <typename VecT>
2216 static __forceinline__ __device__ VecT optixCoopVecExp2(const VecT& vec);
2219 template <typename VecT>
2220 static __forceinline__ __device__ VecT optixCoopVecLog2(const VecT& vec);
2223 template <typename VecT>
2224 static __forceinline__ __device__ VecT optixCoopVecTanh(const VecT& vec);
2229 template <typename VecTOut, typename VecTIn>
2230 static __forceinline__ __device__ VecTOut optixCoopVecCvt(const VecTIn& vec);
2233 template <typename VecT>
2234 static __forceinline__ __device__ VecT optixCoopVecMin(const VecT& vecA, const VecT& vecB);
2237 template <typename VecT>
2238 static __forceinline__ __device__ VecT optixCoopVecMin(const VecT& vecA, typename VecT::value_type B);
2241 template <typename VecT>
2242 static __forceinline__ __device__ VecT optixCoopVecMax(const VecT& vecA, const VecT& vecB);
2245 template <typename VecT>
2246 static __forceinline__ __device__ VecT optixCoopVecMax(const VecT& vecA, typename VecT::value_type B);
2249 template <typename VecT>
2250 static __forceinline__ __device__ VecT optixCoopVecMul(const VecT& vecA, const VecT& vecB);
2253 template <typename VecT>
2254 static __forceinline__ __device__ VecT optixCoopVecAdd(const VecT& vecA, const VecT& vecB);
2257 template <typename VecT>
2258 static __forceinline__ __device__ VecT optixCoopVecSub(const VecT& vecA, const VecT& vecB);
2261 template <typename VecT>
2262 static __forceinline__ __device__ VecT optixCoopVecStep(const VecT& vecA, const VecT& vecB);
2265 template <typename VecT>
2266 static __forceinline__ __device__ VecT optixCoopVecFFMA(const VecT& vecA, const VecT& vecB, const VecT&
vecC);
2267
2332 template <
2333     typename VecTOut,
2334     typename VecTIn,
2335     // For packed element types, the input vector type must be i32 and
2336     // the bits are reinterpreted as one or more of the inputInterpretation type. For
2337     // example, if the input type was OPTIX_COOP_VEC_ELEM_TYPE_INT32 and the inputInterpreation
2338     // value was OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16, then for each element in inputVector, 2 half
2339     // floats will be produced.
2340     OptixCoopVecElemType inputInterpretation,
2341     OptixCoopVecMatrixLayout matrixLayout,
2342     bool transpose,
2343     unsigned int N,
2344     // Note if you are using a packed type, you can have
2345     // situations where not all of the elements in the packed type will be used (e.g. <10
2346     // x __half2>, but K is 19).
2347     unsigned int K,
2348     OptixCoopVecElemType matrixElementType,
2349     // jbigler: note, that I thought about allowing OPTIX_COOP_VEC_ELEM_TYPE_UNKNOWN for
2350     // biasElementType, and take the VecTOut::value_type as the biasElementType, but I
2351     // think it's easy enough for the caller to do that.
2352     OptixCoopVecElemType biasElementType>
2353 static __forceinline__ __device__ VecTOut optixCoopVecMatMul(const VecTIn& inputVector,
2354                                                              CUdeviceptr matrix,  // 64 byte aligned,
Array of KxN elements
2355                                                             unsigned    matrixOffsetInBytes,  // 64 byte
aligned
2356                                                             CUdeviceptr bias,  // 16 byte aligned, Array
```

```
of N elements
2357                                                                    unsigned    biasOffsetInBytes,  // 16 byte
aligned
2358                                                                    unsigned    rowColumnStrideInBytes = 0);
2359
2376 template <typename VecTIn>
2377 static __forceinline__ __device__ void optixCoopVecReduceSumAccumulate(const VecTIn& inputVector,
2378                                                                         CUdeviceptr   outputVector,
2379                                                                         unsigned      offsetInBytes);
2380
2405 template <typename VecTA, typename VecTB, OptixCoopVecMatrixLayout matrixLayout =
OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL>
2406 static __forceinline__ __device__ void optixCoopVecOuterProductAccumulate(const VecTA& vecA,
2407                                                                           const VecTB& vecB,
2408                                                                           CUdeviceptr  outputMatrix,
2409                                                                           unsigned     offsetInBytes,
2410                                                                           unsigned
rowColumnStrideInBytes = 0);
2411
2434 template <unsigned int N, unsigned int K, OptixCoopVecElemType elementType, OptixCoopVecMatrixLayout
layout = OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL, unsigned int rowColumnStrideInBytes = 0>
2435 static __forceinline__ __device__ unsigned int optixCoopVecGetMatrixSize();
2436
2441 template <typename T, unsigned int N>
2442 class OptixCoopVec
2443 {
2444   public:
2445     static const unsigned int size = N;
2446     using value_type               = T;
2447
2448     __forceinline__ __device__ OptixCoopVec() {}
2449     __forceinline__ __device__ OptixCoopVec(const value_type& val)
2450     {
2451         for(unsigned int i = 0; i < size; ++i)
2452             m_data[i]       = val;
2453     }
2454     __forceinline__ __device__ const value_type& operator[](unsigned int index)const { return
m_data[index]; }
2455     __forceinline__ __device__ value_type& operator[](unsigned int index) { return m_data[index]; }
2456
2457     __forceinline__ __device__ const value_type* data()const { return m_data; }
2458     __forceinline__ __device__ value_type* data() { return m_data; }
2459
2460   protected:
2461     value_type m_data[size];
2462 };
2463   // end group optix_device_api
2465
2466 #include "internal/optix_device_impl_coop_vec.h"
2467
2468 #endif //  OPTIX_INCLUDE_COOPERATIVE_VECTOR
2469
2470 #ifdef OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET
2471 #  undef OPTIX_INCLUDE_COOPERATIVE_VECTOR
2472 #  undef OPTIX_INCLUDE_COOPERATIVE_VECTOR_UNSET
2473 #endif
2474
2475
2476 #endif  // OPTIX_OPTIX_DEVICE_H
```

8.13 optix_function_table.h File Reference

Classes

- struct OptixFunctionTable

Macros

- #define OPTIX_ABI_VERSION 105
- #define OPTIX_CONCATENATE_ABI_VERSION(prefix, macro) OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro)
- #define OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro) prefix ## _ ## macro
- #define OPTIX_FUNCTION_TABLE_SYMBOL OPTIX_CONCATENATE_ABI_VERSION(g_optixFunctionTable, OPTIX_ABI_VERSION)

Typedefs

- typedef struct OptixFunctionTable OptixFunctionTable

8.13.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.13.2 Macro Definition Documentation

8.13.2.1 OPTIX_ABI_VERSION

```
#define OPTIX_ABI_VERSION 105
```

The OptiX ABI version. When changing the ABI version make sure you know exactly what you are doing. See apps/optix/exp/functionTable/functionTable.cpp for instructions. See `https://confluence.nvidia.com/display/RAV/ABI+Versions+in+the+Wild` for released ABI versions.

8.14 optix_function_table.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
4 *
5 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
6 * property and proprietary rights in and to this material, related
7 * documentation and any modifications thereto. Any use, reproduction,
8 * disclosure or distribution of this material and related documentation
9 * without an express license agreement from NVIDIA CORPORATION or
10 * its affiliates is strictly prohibited.
11 */
15
16 #ifndef OPTIX_OPTIX_FUNCTION_TABLE_H
17 #define OPTIX_OPTIX_FUNCTION_TABLE_H
18
23 #define OPTIX_ABI_VERSION 105
24
25 #ifndef OPTIX_DEFINE_ABI_VERSION_ONLY
26
27 #include "optix_types.h"
28
29 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
30 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver types must be defined through other
31 // means before including optix headers.
32 #include <cuda.h>
33 #endif
```

```
34
35 #ifdef __cplusplus
36 extern "C" {
37 #endif
38
41
49 typedef struct OptixFunctionTable
50 {
52     //@ {
53
55     const char* (*optixGetErrorName)(OptixResult result);
56
58     const char* (*optixGetErrorString)(OptixResult result);
59
60     //@ }
62     //@ {
63
65     OptixResult (*optixDeviceContextCreate)(CUcontext fromContext, const OptixDeviceContextOptions*
options, OptixDeviceContext* context);
66
68     OptixResult (*optixDeviceContextDestroy)(OptixDeviceContext context);
69
71     OptixResult (*optixDeviceContextGetProperty)(OptixDeviceContext context, OptixDeviceProperty
property, void* value, size_t sizeInBytes);
72
74     OptixResult (*optixDeviceContextSetLogCallback)(OptixDeviceContext context,
75                                                     OptixLogCallback   callbackFunction,
76                                                     void*              callbackData,
77                                                     unsigned int       callbackLevel);
78
80     OptixResult (*optixDeviceContextSetCacheEnabled)(OptixDeviceContext context, int enabled);
81
83     OptixResult (*optixDeviceContextSetCacheLocation)(OptixDeviceContext context, const char* location);
84
86     OptixResult (*optixDeviceContextSetCacheDatabaseSizes)(OptixDeviceContext context, size_t
lowWaterMark, size_t highWaterMark);
87
89     OptixResult (*optixDeviceContextGetCacheEnabled)(OptixDeviceContext context, int* enabled);
90
92     OptixResult (*optixDeviceContextGetCacheLocation)(OptixDeviceContext context, char* location, size_t
locationSize);
93
95     OptixResult (*optixDeviceContextGetCacheDatabaseSizes)(OptixDeviceContext context, size_t*
lowWaterMark, size_t* highWaterMark);
96
97     //@ }
99     //@ {
100
102     OptixResult (*optixModuleCreate)(OptixDeviceContext                 context,
103                                      const OptixModuleCompileOptions*   moduleCompileOptions,
104                                      const OptixPipelineCompileOptions* pipelineCompileOptions,
105                                      const char*                        input,
106                                      size_t                             inputSize,
107                                      char*                              logString,
108                                      size_t*                            logStringSize,
109                                      OptixModule*                       module);
110
112     OptixResult (*optixModuleCreateWithTasks)(OptixDeviceContext                 context,
113                                               const OptixModuleCompileOptions*   moduleCompileOptions,
114                                               const OptixPipelineCompileOptions* pipelineCompileOptions,
115                                               const char*                        input,
116                                               size_t                             inputSize,
117                                               char*                              logString,
118                                               size_t*                            logStringSize,
119                                               OptixModule*                       module,
120                                               OptixTask*                         firstTask);
121
```

```
123     OptixResult (*optixModuleGetCompilationState)(OptixModule module, OptixModuleCompileState* state);
124
126     OptixResult (*optixModuleDestroy)(OptixModule module);
127
129     OptixResult(*optixBuiltinISModuleGet)(OptixDeviceContext                context,
130                                           const OptixModuleCompileOptions*   moduleCompileOptions,
131                                           const OptixPipelineCompileOptions* pipelineCompileOptions,
132                                           const OptixBuiltinISOptions*       builtinISOptions,
133                                           OptixModule*                       builtinModule);
134
135     //@ }
137     //@ {
138
140     OptixResult (*optixTaskExecute)(OptixTask     task,
141                                     OptixTask*    additionalTasks,
142                                     unsigned int  maxNumAdditionalTasks,
143                                     unsigned int* numAdditionalTasksCreated);
144     //@ }
146     //@ {
147
149     OptixResult (*optixProgramGroupCreate)(OptixDeviceContext             context,
150                                            const OptixProgramGroupDesc*    programDescriptions,
151                                            unsigned int                    numProgramGroups,
152                                            const OptixProgramGroupOptions* options,
153                                            char*                           logString,
154                                            size_t*                         logStringSize,
155                                            OptixProgramGroup*              programGroups);
156
158     OptixResult (*optixProgramGroupDestroy)(OptixProgramGroup programGroup);
159
161     OptixResult (*optixProgramGroupGetStackSize)(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes, OptixPipeline pipeline);
162
163     //@ }
165     //@ {
166
168     OptixResult (*optixPipelineCreate)(OptixDeviceContext                context,
169                                        const OptixPipelineCompileOptions* pipelineCompileOptions,
170                                        const OptixPipelineLinkOptions*    pipelineLinkOptions,
171                                        const OptixProgramGroup*           programGroups,
172                                        unsigned int                       numProgramGroups,
173                                        char*                              logString,
174                                        size_t*                            logStringSize,
175                                        OptixPipeline*                     pipeline);
176
178     OptixResult (*optixPipelineDestroy)(OptixPipeline pipeline);
179
181     OptixResult (*optixPipelineSetStackSize)(OptixPipeline pipeline,
182                                              unsigned int  directCallableStackSizeFromTraversal,
183                                              unsigned int  directCallableStackSizeFromState,
184                                              unsigned int  continuationStackSize,
185                                              unsigned int  maxTraversableGraphDepth);
186
187     //@ }
189     //@ {
190
192     OptixResult (*optixAccelComputeMemoryUsage)(OptixDeviceContext            context,
193                                                 const OptixAccelBuildOptions* accelOptions,
194                                                 const OptixBuildInput*        buildInputs,
195                                                 unsigned int                  numBuildInputs,
196                                                 OptixAccelBufferSizes*        bufferSizes);
197
199     OptixResult (*optixAccelBuild)(OptixDeviceContext            context,
200                                    CUstream                      stream,
201                                    const OptixAccelBuildOptions* accelOptions,
202                                    const OptixBuildInput*        buildInputs,
203                                    unsigned int                  numBuildInputs,
```

```
204                                     CUdeviceptr                  tempBuffer,
205                                     size_t                       tempBufferSizeInBytes,
206                                     CUdeviceptr                  outputBuffer,
207                                     size_t                       outputBufferSizeInBytes,
208                                     OptixTraversableHandle*      outputHandle,
209                                     const OptixAccelEmitDesc*    emittedProperties,
210                                     unsigned int                 numEmittedProperties);
211
213     OptixResult (*optixAccelGetRelocationInfo)(OptixDeviceContext context, OptixTraversableHandle
handle, OptixRelocationInfo* info);
214
215
217     OptixResult (*optixCheckRelocationCompatibility)(OptixDeviceContext        context,
218                                                      const OptixRelocationInfo* info,
219                                                      int*                       compatible);
220
222     OptixResult (*optixAccelRelocate)(OptixDeviceContext        context,
223                                       CUstream                  stream,
224                                       const OptixRelocationInfo* info,
225                                       const OptixRelocateInput*  relocateInputs,
226                                       size_t                    numRelocateInputs,
227                                       CUdeviceptr               targetAccel,
228                                       size_t                    targetAccelSizeInBytes,
229                                       OptixTraversableHandle*   targetHandle);
230
231
233     OptixResult (*optixAccelCompact)(OptixDeviceContext      context,
234                                      CUstream                stream,
235                                      OptixTraversableHandle  inputHandle,
236                                      CUdeviceptr             outputBuffer,
237                                      size_t                  outputBufferSizeInBytes,
238                                      OptixTraversableHandle* outputHandle);
239
240     OptixResult (*optixAccelEmitProperty)(OptixDeviceContext        context,
241                                           CUstream                  stream,
242                                           OptixTraversableHandle    handle,
243                                           const OptixAccelEmitDesc* emittedProperty);
244
246     OptixResult (*optixConvertPointerToTraversableHandle)(OptixDeviceContext      onDevice,
247                                                           CUdeviceptr             pointer,
248                                                           OptixTraversableType    traversableType,
249                                                           OptixTraversableHandle* traversableHandle);
250
252     OptixResult (*optixOpacityMicromapArrayComputeMemoryUsage)(OptixDeviceContext
context,
253                                                                const OptixOpacityMicromapArrayBuildInput*
buildInput,
254                                                                   OptixMicromapBufferSizes*
bufferSizes);
255
257     OptixResult (*optixOpacityMicromapArrayBuild)(OptixDeviceContext                        context,
258                                                   CUstream                                  stream,
259                                                   const OptixOpacityMicromapArrayBuildInput* buildInput,
260                                                   const OptixMicromapBuffers*               buffers);
261
263     OptixResult (*optixOpacityMicromapArrayGetRelocationInfo)(OptixDeviceContext   context,
264                                                               CUdeviceptr          opacityMicromapArray,
265                                                                 OptixRelocationInfo* info);
266
268     OptixResult (*optixOpacityMicromapArrayRelocate)(OptixDeviceContext         context,
269                                                      CUstream                   stream,
270                                                      const OptixRelocationInfo* info,
271                                                    CUdeviceptr              targetOpacityMicromapArray,
272                                                      size_t
targetOpacityMicromapArraySizeInBytes);
273
274     OptixResult (*stub1)(void);
```

```
275     OptixResult (*stub2)(void);
276
278     OptixResult (*optixClusterAccelComputeMemoryUsage)(OptixDeviceContext                context,
279                                                        OptixClusterAccelBuildMode         buildMode,
280                                                        const OptixClusterAccelBuildInput* buildInput,
281                                                        OptixAccelBufferSizes*             bufferSizes);
282
284     OptixResult (*optixClusterAccelBuild)(OptixDeviceContext                    context,
285                                           CUstream                              stream,
286                                           const OptixClusterAccelBuildModeDesc* buildModeDesc,
287                                           const OptixClusterAccelBuildInput*    buildInput,
288                                           CUdeviceptr                           argsArray,
289                                           CUdeviceptr                           argsCount,
290                                           unsigned int                          argsStrideInBytes);
291
292     //@ }
294     //@ {
295
297     OptixResult (*optixSbtRecordPackHeader)(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer);
298
300     OptixResult (*optixLaunch)(OptixPipeline                 pipeline,
301                                CUstream                      stream,
302                                CUdeviceptr                   pipelineParams,
303                                size_t                        pipelineParamsSize,
304                                const OptixShaderBindingTable* sbt,
305                                unsigned int                  width,
306                                unsigned int                  height,
307                                unsigned int                  depth);
308
309     //@ }
311     //@ {
312
314     OptixResult (*optixCoopVecMatrixConvert)(OptixDeviceContext            context,
315                                              CUstream                      stream,
316                                              unsigned int                  numNetworks,
317                                              const OptixNetworkDescription* inputNetworkDescription,
318                                              CUdeviceptr                   inputNetworks,
319                                              size_t                        inputNetworkStrideInBytes,
320                                              const OptixNetworkDescription* outputNetworkDescription,
321                                              CUdeviceptr                   outputNetworks,
322                                              size_t                        outputNetworkStrideInBytes);
323
325     OptixResult (*optixCoopVecMatrixComputeSize)(OptixDeviceContext       context,
326                                                  unsigned int             N,
327                                                  unsigned int             K,
328                                                  OptixCoopVecElemType     elementType,
329                                                  OptixCoopVecMatrixLayout layout,
330                                                  size_t                   rowColumnStrideInBytes,
331                                                  size_t*                  sizeInBytes);
332
333     //@ }
335     //@ {
336
338     OptixResult (*optixDenoiserCreate)(OptixDeviceContext context, OptixDenoiserModelKind modelKind,
const OptixDenoiserOptions* options, OptixDenoiser* returnHandle);
339
341     OptixResult (*optixDenoiserDestroy)(OptixDenoiser handle);
342
344     OptixResult (*optixDenoiserComputeMemoryResources)(const OptixDenoiser handle,
345                                                        unsigned int        maximumInputWidth,
346                                                        unsigned int        maximumInputHeight,
347                                                        OptixDenoiserSizes* returnSizes);
348
350     OptixResult (*optixDenoiserSetup)(OptixDenoiser denoiser,
351                                       CUstream      stream,
352                                       unsigned int  inputWidth,
```

```
353                                         unsigned int  inputHeight,
354                                         CUdeviceptr   state,
355                                         size_t        stateSizeInBytes,
356                                         CUdeviceptr   scratch,
357                                         size_t        scratchSizeInBytes);
358
360     OptixResult (*optixDenoiserInvoke)(OptixDenoiser                   denoiser,
361                                         CUstream                       stream,
362                                         const OptixDenoiserParams*     params,
363                                         CUdeviceptr                    denoiserState,
364                                         size_t                         denoiserStateSizeInBytes,
365                                         const OptixDenoiserGuideLayer * guideLayer,
366                                         const OptixDenoiserLayer *     layers,
367                                         unsigned int                   numLayers,
368                                         unsigned int                   inputOffsetX,
369                                         unsigned int                   inputOffsetY,
370                                         CUdeviceptr                    scratch,
371                                         size_t                         scratchSizeInBytes);
372
374     OptixResult (*optixDenoiserComputeIntensity)(OptixDenoiser       handle,
375                                                  CUstream            stream,
376                                                  const OptixImage2D* inputImage,
377                                                  CUdeviceptr         outputIntensity,
378                                                  CUdeviceptr         scratch,
379                                                  size_t              scratchSizeInBytes);
380
382     OptixResult (*optixDenoiserComputeAverageColor)(OptixDenoiser       handle,
383                                                     CUstream            stream,
384                                                     const OptixImage2D* inputImage,
385                                                     CUdeviceptr         outputAverageColor,
386                                                     CUdeviceptr         scratch,
387                                                     size_t              scratchSizeInBytes);
388
390     OptixResult (*optixDenoiserCreateWithUserModel)(OptixDeviceContext context, const void * data, size_t
dataSizeInBytes, OptixDenoiser* returnHandle);
391     //@ }
392
393 } OptixFunctionTable;
394
395 // define global function table variable with ABI specific name.
396 #define OPTIX_CONCATENATE_ABI_VERSION(prefix, macro) OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro)
397 #define OPTIX_CONCATENATE_ABI_VERSION_IMPL(prefix, macro) prefix ## _ ## macro
398 #define OPTIX_FUNCTION_TABLE_SYMBOL OPTIX_CONCATENATE_ABI_VERSION(g_optixFunctionTable,
OPTIX_ABI_VERSION)
399   // end group optix_function_table
401
402 #ifdef __cplusplus
403 }
404 #endif
405
406 #endif /* OPTIX_DEFINE_ABI_VERSION_ONLY */
407
408 #endif /* OPTIX_OPTIX_FUNCTION_TABLE_H */
```

8.15 optix_function_table_definition.h File Reference

Variables

- OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL

8.15.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.16 optix_function_table_definition.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
16 * contributors may be used to endorse or promote products derived from
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
34
35 #ifndef OPTIX_OPTIX_FUNCTION_TABLE_DEFINITION_H
36 #define OPTIX_OPTIX_FUNCTION_TABLE_DEFINITION_H
37
38 #include "optix_function_table.h"
39
40 #ifdef __cplusplus
41 extern "C" {
42 #endif
43
51 OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL;
52   // end group optix_function_table
54
55 #ifdef __cplusplus
56 }
57 #endif
58
59 #endif  // OPTIX_OPTIX_FUNCTION_TABLE_DEFINITION_H
```

8.17 optix_host.h File Reference

Macros

- #define OPTIXAPI

Functions

- OPTIXAPI const char * optixGetErrorName (OptixResult result)

- OPTIXAPI const char * optixGetErrorString (OptixResult result)
- OPTIXAPI OptixResult optixDeviceContextCreate (CUcontext fromContext, const OptixDeviceContextOptions *options, OptixDeviceContext *context)
- OPTIXAPI OptixResult optixDeviceContextDestroy (OptixDeviceContext context)
- OPTIXAPI OptixResult optixDeviceContextGetProperty (OptixDeviceContext context, OptixDeviceProperty property, void *value, size_t sizeInBytes)
- OPTIXAPI OptixResult optixDeviceContextSetLogCallback (OptixDeviceContext context, OptixLogCallback callbackFunction, void *callbackData, unsigned int callbackLevel)
- OPTIXAPI OptixResult optixDeviceContextSetCacheEnabled (OptixDeviceContext context, int enabled)
- OPTIXAPI OptixResult optixDeviceContextSetCacheLocation (OptixDeviceContext context, const char *location)
- OPTIXAPI OptixResult optixDeviceContextSetCacheDatabaseSizes (OptixDeviceContext context, size_t lowWaterMark, size_t highWaterMark)
- OPTIXAPI OptixResult optixDeviceContextGetCacheEnabled (OptixDeviceContext context, int *enabled)
- OPTIXAPI OptixResult optixDeviceContextGetCacheLocation (OptixDeviceContext context, char *location, size_t locationSize)
- OPTIXAPI OptixResult optixDeviceContextGetCacheDatabaseSizes (OptixDeviceContext context, size_t *lowWaterMark, size_t *highWaterMark)
- OPTIXAPI OptixResult optixPipelineCreate (OptixDeviceContext context, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixPipelineLinkOptions *pipelineLinkOptions, const OptixProgramGroup *programGroups, unsigned int numProgramGroups, char *logString, size_t *logStringSize, OptixPipeline *pipeline)
- OPTIXAPI OptixResult optixPipelineDestroy (OptixPipeline pipeline)
- OPTIXAPI OptixResult optixPipelineSetStackSize (OptixPipeline pipeline, unsigned int directCallableStackSizeFromTraversal, unsigned int directCallableStackSizeFromState, unsigned int continuationStackSize, unsigned int maxTraversableGraphDepth)
- OPTIXAPI OptixResult optixModuleCreate (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module)
- OPTIXAPI OptixResult optixModuleCreateWithTasks (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const char *input, size_t inputSize, char *logString, size_t *logStringSize, OptixModule *module, OptixTask *firstTask)
- OPTIXAPI OptixResult optixModuleGetCompilationState (OptixModule module, OptixModuleCompileState *state)
- OPTIXAPI OptixResult optixModuleDestroy (OptixModule module)
- OPTIXAPI OptixResult optixBuiltinISModuleGet (OptixDeviceContext context, const OptixModuleCompileOptions *moduleCompileOptions, const OptixPipelineCompileOptions *pipelineCompileOptions, const OptixBuiltinISOptions *builtinISOptions, OptixModule *builtinModule)
- OPTIXAPI OptixResult optixTaskExecute (OptixTask task, OptixTask *additionalTasks, unsigned int maxNumAdditionalTasks, unsigned int *numAdditionalTasksCreated)
- OPTIXAPI OptixResult optixProgramGroupGetStackSize (OptixProgramGroup programGroup, OptixStackSizes *stackSizes, OptixPipeline pipeline)
- OPTIXAPI OptixResult optixProgramGroupCreate (OptixDeviceContext context, const OptixProgramGroupDesc *programDescriptions, unsigned int numProgramGroups, const OptixProgramGroupOptions *options, char *logString, size_t *logStringSize, OptixProgramGroup *programGroups)
- OPTIXAPI OptixResult optixProgramGroupDestroy (OptixProgramGroup programGroup)

- OPTIXAPI OptixResult optixSbtRecordPackHeader (OptixProgramGroup programGroup, void *sbtRecordHeaderHostPointer)
- OPTIXAPI OptixResult optixLaunch (OptixPipeline pipeline, CUstream stream, CUdeviceptr pipelineParams, size_t pipelineParamsSize, const OptixShaderBindingTable *sbt, unsigned int width, unsigned int height, unsigned int depth)
- OPTIXAPI OptixResult optixAccelComputeMemoryUsage (OptixDeviceContext context, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, OptixAccelBufferSizes *bufferSizes)
- OPTIXAPI OptixResult optixAccelBuild (OptixDeviceContext context, CUstream stream, const OptixAccelBuildOptions *accelOptions, const OptixBuildInput *buildInputs, unsigned int numBuildInputs, CUdeviceptr tempBuffer, size_t tempBufferSizeInBytes, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle, const OptixAccelEmitDesc *emittedProperties, unsigned int numEmittedProperties)
- OPTIXAPI OptixResult optixAccelGetRelocationInfo (OptixDeviceContext context, OptixTraversableHandle handle, OptixRelocationInfo *info)
- OPTIXAPI OptixResult optixCheckRelocationCompatibility (OptixDeviceContext context, const OptixRelocationInfo *info, int *compatible)
- OPTIXAPI OptixResult optixAccelRelocate (OptixDeviceContext context, CUstream stream, const OptixRelocationInfo *info, const OptixRelocateInput *relocateInputs, size_t numRelocateInputs, CUdeviceptr targetAccel, size_t targetAccelSizeInBytes, OptixTraversableHandle *targetHandle)
- OPTIXAPI OptixResult optixAccelCompact (OptixDeviceContext context, CUstream stream, OptixTraversableHandle inputHandle, CUdeviceptr outputBuffer, size_t outputBufferSizeInBytes, OptixTraversableHandle *outputHandle)
- OPTIXAPI OptixResult optixAccelEmitProperty (OptixDeviceContext context, CUstream stream, OptixTraversableHandle handle, const OptixAccelEmitDesc *emittedProperty)
- OPTIXAPI OptixResult optixConvertPointerToTraversableHandle (OptixDeviceContext onDevice, CUdeviceptr pointer, OptixTraversableType traversableType, OptixTraversableHandle *traversableHandle)
- OPTIXAPI OptixResult optixOpacityMicromapArrayComputeMemoryUsage (OptixDeviceContext context, const OptixOpacityMicromapArrayBuildInput *buildInput, OptixMicromapBufferSizes *bufferSizes)
- OPTIXAPI OptixResult optixOpacityMicromapArrayBuild (OptixDeviceContext context, CUstream stream, const OptixOpacityMicromapArrayBuildInput *buildInput, const OptixMicromapBuffers *buffers)
- OPTIXAPI OptixResult optixOpacityMicromapArrayGetRelocationInfo (OptixDeviceContext context, CUdeviceptr opacityMicromapArray, OptixRelocationInfo *info)
- OPTIXAPI OptixResult optixOpacityMicromapArrayRelocate (OptixDeviceContext context, CUstream stream, const OptixRelocationInfo *info, CUdeviceptr targetOpacityMicromapArray, size_t targetOpacityMicromapArraySizeInBytes)
- OPTIXAPI OptixResult optixClusterAccelComputeMemoryUsage (OptixDeviceContext context, OptixClusterAccelBuildMode buildMode, const OptixClusterAccelBuildInput *buildInput, OptixAccelBufferSizes *bufferSizes)
- OPTIXAPI OptixResult optixClusterAccelBuild (OptixDeviceContext context, CUstream stream, const OptixClusterAccelBuildModeDesc *buildModeDesc, const OptixClusterAccelBuildInput *buildInput, CUdeviceptr argsArray, CUdeviceptr argsCount, unsigned int argsStrideInBytes)
- OPTIXAPI OptixResult optixCoopVecMatrixConvert (OptixDeviceContext context, CUstream stream, unsigned int numNetworks, const OptixNetworkDescription *inputNetworkDescription, CUdeviceptr inputNetworks, size_t inputNetworkStrideInBytes, const OptixNetworkDescription *outputNetworkDescription, CUdeviceptr outputNetworks, size_t outputNetworkStrideInBytes)
- OPTIXAPI OptixResult optixCoopVecMatrixComputeSize (OptixDeviceContext context, unsigned int N, unsigned int K, OptixCoopVecElemType elementType, OptixCoopVecMatrixLayout layout, size_t rowColumnStrideInBytes, size_t *sizeInBytes)

- OPTIXAPI OptixResult optixDenoiserCreate (OptixDeviceContext context, OptixDenoiserModelKind modelKind, const OptixDenoiserOptions *options, OptixDenoiser *denoiser)
- OPTIXAPI OptixResult optixDenoiserCreateWithUserModel (OptixDeviceContext context, const void *userData, size_t userDataSizeInBytes, OptixDenoiser *denoiser)
- OPTIXAPI OptixResult optixDenoiserDestroy (OptixDenoiser denoiser)
- OPTIXAPI OptixResult optixDenoiserComputeMemoryResources (const OptixDenoiser denoiser, unsigned int outputWidth, unsigned int outputHeight, OptixDenoiserSizes *returnSizes)
- OPTIXAPI OptixResult optixDenoiserSetup (OptixDenoiser denoiser, CUstream stream, unsigned int inputWidth, unsigned int inputHeight, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OPTIXAPI OptixResult optixDenoiserInvoke (OptixDenoiser denoiser, CUstream stream, const OptixDenoiserParams *params, CUdeviceptr denoiserState, size_t denoiserStateSizeInBytes, const OptixDenoiserGuideLayer *guideLayer, const OptixDenoiserLayer *layers, unsigned int numLayers, unsigned int inputOffsetX, unsigned int inputOffsetY, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OPTIXAPI OptixResult optixDenoiserComputeIntensity (OptixDenoiser denoiser, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputIntensity, CUdeviceptr scratch, size_t scratchSizeInBytes)
- OPTIXAPI OptixResult optixDenoiserComputeAverageColor (OptixDenoiser denoiser, CUstream stream, const OptixImage2D *inputImage, CUdeviceptr outputAverageColor, CUdeviceptr scratch, size_t scratchSizeInBytes)

Variables

- int pointerType

8.17.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

OptiX host include file – includes the host api if compiling host code. For the math library routines include optix_math.h

8.17.2 Macro Definition Documentation

8.17.2.1 OPTIXAPI

```
#define OPTIXAPI
```

Mixing multiple SDKs in a single application will result in symbol collisions. To enable different compilation units to use different SDKs, use OPTIX_ENABLE_SDK_MIXING.

8.18 optix_host.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2010 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
4 *
5 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
6 * property and proprietary rights in and to this material, related
```

```
7 * documentation and any modifications thereto. Any use, reproduction,
8 * disclosure or distribution of this material and related documentation
9 * without an express license agreement from NVIDIA CORPORATION or
10 * its affiliates is strictly prohibited.
11 */
18
19 #ifndef OPTIX_OPTIX_HOST_H
20 #define OPTIX_OPTIX_HOST_H
21
24 #ifndef OPTIXAPI
25 # ifdef OPTIX_ENABLE_SDK_MIXING
26 #   define OPTIXAPI static
27 # else  // OPTIX_ENABLE_SDK_MIXING
28 #   ifdef __cplusplus
29 #     define OPTIXAPI extern "C"
30 #   else  // __cplusplus
31 #     define OPTIXAPI
32 #   endif  // __cplusplus
33 # endif  // OPTIX_ENABLE_SDK_MIXING
34 #endif  // OPTIXAPI
35
36 #include "optix_types.h"
37 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
38 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver types must be defined through other
39 // means before including optix headers.
40 #include <cuda.h>
41 #endif
42
43 #ifdef NV_MODULE_OPTIX
44 // This is a mechanism to include <g_nvconfig.h> in driver builds only and translate any nvconfig macro to
a custom OPTIX-specific macro, that can also be used in SDK builds/installs
45 #include <exp/misc/optix_nvconfig_translate.h>  // includes <g_nvconfig.h>
46 #endif  // NV_MODULE_OPTIX
47
48
51
55
66 OPTIXAPI const char* optixGetErrorName(OptixResult result);
67
78 OPTIXAPI const char* optixGetErrorString(OptixResult result);
79
84
103 OPTIXAPI OptixResult optixDeviceContextCreate(CUcontext fromContext, const OptixDeviceContextOptions*
options, OptixDeviceContext* context);
104
113 OPTIXAPI OptixResult optixDeviceContextDestroy(OptixDeviceContext context);
114
121 OPTIXAPI OptixResult optixDeviceContextGetProperty(OptixDeviceContext context, OptixDeviceProperty
property, void* value, size_t sizeInBytes);
122
136 OPTIXAPI OptixResult optixDeviceContextSetLogCallback(OptixDeviceContext context,
137                                                       OptixLogCallback   callbackFunction,
138                                                       void*              callbackData,
139                                                       unsigned int       callbackLevel);
140
159 OPTIXAPI OptixResult optixDeviceContextSetCacheEnabled(OptixDeviceContext context, int enabled);
160
181 OPTIXAPI OptixResult optixDeviceContextSetCacheLocation(OptixDeviceContext context, const char*
location);
182
210 OPTIXAPI OptixResult optixDeviceContextSetCacheDatabaseSizes(OptixDeviceContext context, size_t
lowWaterMark, size_t highWaterMark);
211
216 OPTIXAPI OptixResult optixDeviceContextGetCacheEnabled(OptixDeviceContext context, int* enabled);
223 OPTIXAPI OptixResult optixDeviceContextGetCacheLocation(OptixDeviceContext context, char* location,
size_t locationSize);
224
```

```
232 OPTIXAPI OptixResult optixDeviceContextGetCacheDatabaseSizes(OptixDeviceContext context, size_t*
lowWaterMark, size_t* highWaterMark);
233
238
262 OPTIXAPI OptixResult optixPipelineCreate(OptixDeviceContext                 context,
263                                          const OptixPipelineCompileOptions* pipelineCompileOptions,
264                                          const OptixPipelineLinkOptions*    pipelineLinkOptions,
265                                          const OptixProgramGroup*           programGroups,
266                                          unsigned int                       numProgramGroups,
267                                          char*                              logString,
268                                          size_t*                            logStringSize,
269                                          OptixPipeline*                     pipeline);
270
272 OPTIXAPI OptixResult optixPipelineDestroy(OptixPipeline pipeline);
273
296 OPTIXAPI OptixResult optixPipelineSetStackSize(OptixPipeline pipeline,
297                                                unsigned int  directCallableStackSizeFromTraversal,
298                                                unsigned int  directCallableStackSizeFromState,
299                                                unsigned int  continuationStackSize,
300                                                unsigned int  maxTraversableGraphDepth);
301
306
336 OPTIXAPI OptixResult optixModuleCreate(OptixDeviceContext                 context,
337                                        const OptixModuleCompileOptions*   moduleCompileOptions,
338                                        const OptixPipelineCompileOptions* pipelineCompileOptions,
339                                        const char*                        input,
340                                        size_t                             inputSize,
341                                        char*                              logString,
342                                        size_t*                            logStringSize,
343                                        OptixModule*                       module);
344
385 OPTIXAPI OptixResult optixModuleCreateWithTasks(OptixDeviceContext                 context,
386                                                 const OptixModuleCompileOptions*   moduleCompileOptions,
387                                                 const OptixPipelineCompileOptions* pipelineCompileOptions,
388                                                 const char*                        input,
389                                                 size_t                             inputSize,
390                                                 char*                              logString,
391                                                 size_t*                            logStringSize,
392                                                 OptixModule*                       module,
393                                                 OptixTask*                         firstTask);
394
401 OPTIXAPI OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState* state);
402
408 OPTIXAPI OptixResult optixModuleDestroy(OptixModule module);
409
413 OPTIXAPI OptixResult optixBuiltinISModuleGet(OptixDeviceContext                 context,
414                                              const OptixModuleCompileOptions*   moduleCompileOptions,
415                                              const OptixPipelineCompileOptions* pipelineCompileOptions,
416                                              const OptixBuiltinISOptions*       builtinISOptions,
417                                              OptixModule*                       builtinModule);
418
423
441 OPTIXAPI OptixResult optixTaskExecute(OptixTask     task,
442                                       OptixTask*    additionalTasks,
443                                       unsigned int  maxNumAdditionalTasks,
444                                       unsigned int* numAdditionalTasksCreated);
445
450
459 OPTIXAPI OptixResult optixProgramGroupGetStackSize(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes, OptixPipeline pipeline);
460
486 OPTIXAPI OptixResult optixProgramGroupCreate(OptixDeviceContext             context,
487                                              const OptixProgramGroupDesc*    programDescriptions,
488                                              unsigned int                    numProgramGroups,
489                                              const OptixProgramGroupOptions* options,
490                                              char*                           logString,
491                                              size_t*                         logStringSize,
```

```
492                                             OptixProgramGroup*              programGroups);
493
495 OPTIXAPI OptixResult optixProgramGroupDestroy(OptixProgramGroup programGroup);
496
499 OPTIXAPI OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer);
500
505
532 OPTIXAPI OptixResult optixLaunch(OptixPipeline                 pipeline,
533                                  CUstream                      stream,
534                                  CUdeviceptr                   pipelineParams,
535                                  size_t                        pipelineParamsSize,
536                                  const OptixShaderBindingTable* sbt,
537                                  unsigned int                  width,
538                                  unsigned int                  height,
539                                  unsigned int                  depth);
540
545
551 OPTIXAPI OptixResult optixAccelComputeMemoryUsage(OptixDeviceContext           context,
552                                                   const OptixAccelBuildOptions* accelOptions,
553                                                   const OptixBuildInput*        buildInputs,
554                                                   unsigned int                  numBuildInputs,
555                                                   OptixAccelBufferSizes*        bufferSizes);
556
569 OPTIXAPI OptixResult optixAccelBuild(OptixDeviceContext            context,
570                                      CUstream                      stream,
571                                      const OptixAccelBuildOptions* accelOptions,
572                                      const OptixBuildInput*        buildInputs,
573                                      unsigned int                  numBuildInputs,
574                                      CUdeviceptr                   tempBuffer,
575                                      size_t                        tempBufferSizeInBytes,
576                                      CUdeviceptr                   outputBuffer,
577                                      size_t                        outputBufferSizeInBytes,
578                                      OptixTraversableHandle*       outputHandle,
579                                      const OptixAccelEmitDesc*     emittedProperties,
580                                      unsigned int                  numEmittedProperties);
581
599 OPTIXAPI OptixResult optixAccelGetRelocationInfo(OptixDeviceContext context, OptixTraversableHandle
handle, OptixRelocationInfo* info);
600
612 OPTIXAPI OptixResult optixCheckRelocationCompatibility(OptixDeviceContext context, const
OptixRelocationInfo* info, int* compatible);
613
651 OPTIXAPI OptixResult optixAccelRelocate(OptixDeviceContext         context,
652                                         CUstream                    stream,
653                                         const OptixRelocationInfo* info,
654                                         const OptixRelocateInput*  relocateInputs,
655                                         size_t                     numRelocateInputs,
656                                         CUdeviceptr                targetAccel,
657                                         size_t                     targetAccelSizeInBytes,
658                                         OptixTraversableHandle*    targetHandle);
659
677 OPTIXAPI OptixResult optixAccelCompact(OptixDeviceContext      context,
678                                        CUstream                stream,
679                                        OptixTraversableHandle  inputHandle,
680                                        CUdeviceptr             outputBuffer,
681                                        size_t                  outputBufferSizeInBytes,
682                                        OptixTraversableHandle* outputHandle);
683
692 OPTIXAPI OptixResult optixAccelEmitProperty(OptixDeviceContext        context,
693                                             CUstream                  stream,
694                                             OptixTraversableHandle    handle,
695                                             const OptixAccelEmitDesc* emittedProperty);
696
701 OPTIXAPI OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext      onDevice,
702                                                             CUdeviceptr             pointer,
703                                                             OptixTraversableType    traversableType,
```

```
704                                                         OptixTraversableHandle* traversableHandle);
705
706
712 OPTIXAPI OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext
context,
713                                                           const OptixOpacityMicromapArrayBuildInput*
buildInput,
714                                                             OptixMicromapBufferSizes* bufferSizes);
715
738 OPTIXAPI OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext                       context,
739                                                     CUstream                                  stream,
740                                                 const OptixOpacityMicromapArrayBuildInput* buildInput,
741                                                   const OptixMicromapBuffers*                buffers);
742
758 OPTIXAPI OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext   context,
759                                                               CUdeviceptr          opacityMicromapArray,
760                                                                 OptixRelocationInfo* info);
761
788 OPTIXAPI OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext         context,
789                                                        CUstream                   stream,
790                                                        const OptixRelocationInfo* info,
791                                                    CUdeviceptr                targetOpacityMicromapArray,
792                                                        size_t
targetOpacityMicromapArraySizeInBytes);
793
794
805 OPTIXAPI OptixResult optixClusterAccelComputeMemoryUsage(OptixDeviceContext                 context,
806                                                          OptixClusterAccelBuildMode         buildMode,
807                                                          const OptixClusterAccelBuildInput* buildInput,
808                                                        OptixAccelBufferSizes*             bufferSizes);
809
833 OPTIXAPI OptixResult optixClusterAccelBuild(OptixDeviceContext                    context,
834                                             CUstream                               stream,
835                                             const OptixClusterAccelBuildModeDesc* buildModeDesc,
836                                             const OptixClusterAccelBuildInput*    buildInput,
837                                             CUdeviceptr                            argsArray,
838                                             CUdeviceptr                            argsCount,
839                                             unsigned int                           argsStrideInBytes);
840
841
846
863
864 // There is need to have a driver side function to convert from row/column major layout to optimal layout
865 //
866 // An example of this for VK can be found here:
867 //
https://p4sw-swarm.nvidia.com/files/sw/dev/gpu_drv/bugfix_main/drivers/OpenGL/vulkan/commands/transfer/vktrans
868 //
869 // See ConvertCooperativeVectorMatrixLayoutNV which calls ComputeMatrixOffset.
870
871
881 OPTIXAPI OptixResult optixCoopVecMatrixConvert(OptixDeviceContext             context,
882                                                CUstream                        stream,
883                                                unsigned int                    numNetworks,
884                                                const OptixNetworkDescription* inputNetworkDescription,
885                                                CUdeviceptr                     inputNetworks,
886                                               size_t                          inputNetworkStrideInBytes,
887                                                const OptixNetworkDescription* outputNetworkDescription,
888                                                CUdeviceptr                     outputNetworks,
889                                              size_t                         outputNetworkStrideInBytes);
890
891 // optional future work
892 int pointerType;  // discriminates array of pointers or arrays of structs
893
894
908 OPTIXAPI OptixResult optixCoopVecMatrixComputeSize(OptixDeviceContext       context,
909                                                    unsigned int             N,
```

```
910                                                 unsigned int             K,
911                                                 OptixCoopVecElemType     elementType,
912                                                 OptixCoopVecMatrixLayout layout,
913                                                 size_t                   rowColumnStrideInBytes,
914                                                 size_t*                  sizeInBytes);
915
916
921
933 OPTIXAPI OptixResult optixDenoiserCreate(OptixDeviceContext          context,
934                                           OptixDenoiserModelKind      modelKind,
935                                           const OptixDenoiserOptions* options,
936                                           OptixDenoiser*              denoiser);
937
950 OPTIXAPI OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context,
951                                                       const void*        userData,
952                                                       size_t             userDataSizeInBytes,
953                                                       OptixDenoiser*     denoiser);
954
956 OPTIXAPI OptixResult optixDenoiserDestroy(OptixDenoiser denoiser);
957
977 OPTIXAPI OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser denoiser,
978                                                          unsigned int        outputWidth,
979                                                          unsigned int        outputHeight,
980                                                          OptixDenoiserSizes* returnSizes);
981
998 OPTIXAPI OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
999                                         CUstream      stream,
1000                                         unsigned int  inputWidth,
1001                                         unsigned int  inputHeight,
1002                                         CUdeviceptr   denoiserState,
1003                                         size_t        denoiserStateSizeInBytes,
1004                                         CUdeviceptr   scratch,
1005                                         size_t        scratchSizeInBytes);
1006
1072 OPTIXAPI OptixResult optixDenoiserInvoke(OptixDenoiser                 denoiser,
1073                                          CUstream                      stream,
1074                                          const OptixDenoiserParams*    params,
1075                                          CUdeviceptr                   denoiserState,
1076                                          size_t                        denoiserStateSizeInBytes,
1077                                          const OptixDenoiserGuideLayer* guideLayer,
1078                                          const OptixDenoiserLayer*     layers,
1079                                          unsigned int                  numLayers,
1080                                          unsigned int                  inputOffsetX,
1081                                          unsigned int                  inputOffsetY,
1082                                          CUdeviceptr                   scratch,
1083                                          size_t                        scratchSizeInBytes);
1084
1108 OPTIXAPI OptixResult optixDenoiserComputeIntensity(OptixDenoiser       denoiser,
1109                                                    CUstream            stream,
1110                                                    const OptixImage2D* inputImage,
1111                                                    CUdeviceptr         outputIntensity,
1112                                                    CUdeviceptr         scratch,
1113                                                    size_t              scratchSizeInBytes);
1114
1129 OPTIXAPI OptixResult optixDenoiserComputeAverageColor(OptixDenoiser       denoiser,
1130                                                       CUstream            stream,
1131                                                       const OptixImage2D* inputImage,
1132                                                       CUdeviceptr         outputAverageColor,
1133                                                       CUdeviceptr         scratch,
1134                                                       size_t              scratchSizeInBytes);
1135
1137
1138 #include "optix_function_table.h"
1139
1140 #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.

Parameters

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 * SPDX-FileCopyrightText: Copyright (c) 2022 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
16 * contributors may be used to endorse or promote products derived from
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
39 #ifndef OPTIX_OPTIX_MICROMAP_H
40 #define OPTIX_OPTIX_MICROMAP_H
41
42 #if !defined(OPTIX_DONT_INCLUDE_CUDA)
43 // If OPTIX_DONT_INCLUDE_CUDA is defined, cuda driver type float2 must be defined through other
44 // means before including optix headers.
45 #include <vector_types.h>
46 #endif
47 #include "internal/optix_micromap_impl.h"
48
58 OPTIX_MICROMAP_INLINE_FUNC void optixMicromapIndexToBaseBarycentrics(unsigned int micromapTriangleIndex,
59                                                                       unsigned int subdivisionLevel,
60                                                                       float2&      baseBarycentrics0,
61                                                                       float2&      baseBarycentrics1,
62                                                                       float2&      baseBarycentrics2)
63 {
64     optix_impl::micro2bary(micromapTriangleIndex, subdivisionLevel, baseBarycentrics0, baseBarycentrics1,
baseBarycentrics2);
65 }
66
70 OPTIX_MICROMAP_INLINE_FUNC float2 optixBaseBarycentricsToMicroBarycentrics(float2 baseBarycentrics,
71                                                                            float2 microVertexBaseBarycentrics[3])
72 {
```

```
73     return optix_impl::base2micro(baseBarycentrics, microVertexBaseBarycentrics);
74 }
75
76 #endif  // OPTIX_OPTIX_MICROMAP_H
```

8.21 optix_stack_size.h File Reference

Functions

- OptixResult optixUtilAccumulateStackSizes (OptixProgramGroup programGroup, OptixStackSizes *stackSizes, OptixPipeline pipeline)
- OptixResult optixUtilComputeStackSizes (const OptixStackSizes *stackSizes, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepth, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesDCSplit (const OptixStackSizes *stackSizes, unsigned int dssDCFromTraversal, unsigned int dssDCFromState, unsigned int maxTraceDepth, unsigned int maxCCDepth, unsigned int maxDCDepthFromTraversal, unsigned int maxDCDepthFromState, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesCssCCTree (const OptixStackSizes *stackSizes, unsigned int cssCCTree, unsigned int maxTraceDepth, unsigned int maxDCDepth, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize)
- OptixResult optixUtilComputeStackSizesSimplePathTracer (OptixProgramGroup programGroupRG, OptixProgramGroup programGroupMS1, const OptixProgramGroup *programGroupCH1, unsigned int programGroupCH1Count, OptixProgramGroup programGroupMS2, const OptixProgramGroup *programGroupCH2, unsigned int programGroupCH2Count, unsigned int *directCallableStackSizeFromTraversal, unsigned int *directCallableStackSizeFromState, unsigned int *continuationStackSize, OptixPipeline pipeline)

8.21.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.22 optix_stack_size.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
16 * contributors may be used to endorse or promote products derived from
```

```
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
34
35 #ifndef OPTIX_OPTIX_STACK_SIZE_H
36 #define OPTIX_OPTIX_STACK_SIZE_H
37
38 #include "optix.h"
39
40 #include <algorithm>
41 #include <cstring>
42
43 #ifdef __cplusplus
44 extern "C" {
45 #endif
46
56 inline OptixResult optixUtilAccumulateStackSizes(OptixProgramGroup programGroup, OptixStackSizes*
stackSizes, OptixPipeline pipeline)
57 {
58     if(!stackSizes)
59         return OPTIX_ERROR_INVALID_VALUE;
60
61     OptixStackSizes localStackSizes;
62     OptixResult     result = optixProgramGroupGetStackSize(programGroup, &localStackSizes, pipeline);
63     if(result != OPTIX_SUCCESS)
64         return result;
65
66     stackSizes->cssRG = std::max(stackSizes->cssRG, localStackSizes.cssRG);
67     stackSizes->cssMS = std::max(stackSizes->cssMS, localStackSizes.cssMS);
68     stackSizes->cssCH = std::max(stackSizes->cssCH, localStackSizes.cssCH);
69     stackSizes->cssAH = std::max(stackSizes->cssAH, localStackSizes.cssAH);
70     stackSizes->cssIS = std::max(stackSizes->cssIS, localStackSizes.cssIS);
71     stackSizes->cssCC = std::max(stackSizes->cssCC, localStackSizes.cssCC);
72     stackSizes->dssDC = std::max(stackSizes->dssDC, localStackSizes.dssDC);
73
74     return OPTIX_SUCCESS;
75 }
76
90 inline OptixResult optixUtilComputeStackSizes(const OptixStackSizes* stackSizes,
91                                               unsigned int           maxTraceDepth,
92                                               unsigned int           maxCCDepth,
93                                               unsigned int           maxDCDepth,
94                                              unsigned int*          directCallableStackSizeFromTraversal,
95                                               unsigned int*          directCallableStackSizeFromState,
96                                               unsigned int*          continuationStackSize)
97 {
98     if(!stackSizes)
99         return OPTIX_ERROR_INVALID_VALUE;
100
101     const unsigned int cssRG = stackSizes->cssRG;
102     const unsigned int cssMS = stackSizes->cssMS;
103     const unsigned int cssCH = stackSizes->cssCH;
104     const unsigned int cssAH = stackSizes->cssAH;
105     const unsigned int cssIS = stackSizes->cssIS;
106     const unsigned int cssCC = stackSizes->cssCC;
107     const unsigned int dssDC = stackSizes->dssDC;
```

```
108
109     if(directCallableStackSizeFromTraversal)
110         *directCallableStackSizeFromTraversal = maxDCDepth * dssDC;
111     if(directCallableStackSizeFromState)
112         *directCallableStackSizeFromState = maxDCDepth * dssDC;
113
114     // upper bound on continuation stack used by call trees of continuation callables
115     unsigned int cssCCTree = maxCCDepth * cssCC;
116
117     // upper bound on continuation stack used by CH or MS programs including the call tree of
118     // continuation callables
119     unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
120
121     // clang-format off
122     if(continuationStackSize)
123         *continuationStackSize
124             = cssRG + cssCCTree
125             + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
126             + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
127     // clang-format on
128
129     return OPTIX_SUCCESS;
130 }
131
155 inline OptixResult optixUtilComputeStackSizesDCSplit(const OptixStackSizes* stackSizes,
156                                                      unsigned int           dssDCFromTraversal,
157                                                      unsigned int           dssDCFromState,
158                                                      unsigned int           maxTraceDepth,
159                                                      unsigned int           maxCCDepth,
160                                                      unsigned int           maxDCDepthFromTraversal,
161                                                      unsigned int           maxDCDepthFromState,
162                                                      unsigned int*
directCallableStackSizeFromTraversal,
163                                                  unsigned int*         directCallableStackSizeFromState,
164                                                      unsigned int*          continuationStackSize)
165 {
166     if(!stackSizes)
167         return OPTIX_ERROR_INVALID_VALUE;
168
169     const unsigned int cssRG = stackSizes->cssRG;
170     const unsigned int cssMS = stackSizes->cssMS;
171     const unsigned int cssCH = stackSizes->cssCH;
172     const unsigned int cssAH = stackSizes->cssAH;
173     const unsigned int cssIS = stackSizes->cssIS;
174     const unsigned int cssCC = stackSizes->cssCC;
175     // use dssDCFromTraversal and dssDCFromState instead of stackSizes->dssDC
176
177     if(directCallableStackSizeFromTraversal)
178         *directCallableStackSizeFromTraversal = maxDCDepthFromTraversal * dssDCFromTraversal;
179     if(directCallableStackSizeFromState)
180         *directCallableStackSizeFromState = maxDCDepthFromState * dssDCFromState;
181
182     // upper bound on continuation stack used by call trees of continuation callables
183     unsigned int cssCCTree = maxCCDepth * cssCC;
184
185     // upper bound on continuation stack used by CH or MS programs including the call tree of
186     // continuation callables
187     unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
188
189     // clang-format off
190     if(continuationStackSize)
191         *continuationStackSize
192             = cssRG + cssCCTree
193             + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
194             + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
195     // clang-format on
196
```

```
197     return OPTIX_SUCCESS;
198 }
199
216 inline OptixResult optixUtilComputeStackSizesCssCCTree(const OptixStackSizes* stackSizes,
217                                                         unsigned int           cssCCTree,
218                                                         unsigned int           maxTraceDepth,
219                                                         unsigned int           maxDCDepth,
220                                                         unsigned int*
directCallableStackSizeFromTraversal,
221                                                 unsigned int*         directCallableStackSizeFromState,
222                                                         unsigned int*          continuationStackSize)
223 {
224     if(!stackSizes)
225         return OPTIX_ERROR_INVALID_VALUE;
226
227     const unsigned int cssRG = stackSizes->cssRG;
228     const unsigned int cssMS = stackSizes->cssMS;
229     const unsigned int cssCH = stackSizes->cssCH;
230     const unsigned int cssAH = stackSizes->cssAH;
231     const unsigned int cssIS = stackSizes->cssIS;
232     // use cssCCTree instead of stackSizes->cssCC and maxCCDepth
233     const unsigned int dssDC = stackSizes->dssDC;
234
235     if(directCallableStackSizeFromTraversal)
236         *directCallableStackSizeFromTraversal = maxDCDepth * dssDC;
237     if(directCallableStackSizeFromState)
238         *directCallableStackSizeFromState = maxDCDepth * dssDC;
239
240     // upper bound on continuation stack used by CH or MS programs including the call tree of
241     // continuation callables
242     unsigned int cssCHOrMSPlusCCTree = std::max(cssCH, cssMS) + cssCCTree;
243
244     // clang-format off
245     if(continuationStackSize)
246         *continuationStackSize
247             = cssRG + cssCCTree
248             + (std::max(maxTraceDepth, 1u) - 1) * cssCHOrMSPlusCCTree
249             + std::min(maxTraceDepth, 1u) * std::max(cssCHOrMSPlusCCTree, cssIS + cssAH);
250     // clang-format on
251
252     return OPTIX_SUCCESS;
253 }
254
270 inline OptixResult optixUtilComputeStackSizesSimplePathTracer(OptixProgramGroup        programGroupRG,
271                                                               OptixProgramGroup        programGroupMS1,
272                                                               const OptixProgramGroup* programGroupCH1,
273                                                         unsigned int            programGroupCH1Count,
274                                                               OptixProgramGroup        programGroupMS2,
275                                                               const OptixProgramGroup* programGroupCH2,
276                                                         unsigned int            programGroupCH2Count,
277                                                               unsigned int*
directCallableStackSizeFromTraversal,
278                                                        unsigned int* directCallableStackSizeFromState,
279                                                               unsigned int* continuationStackSize,
280                                                               OptixPipeline pipeline)
281 {
282     if(!programGroupCH1 && (programGroupCH1Count > 0))
283         return OPTIX_ERROR_INVALID_VALUE;
284     if(!programGroupCH2 && (programGroupCH2Count > 0))
285         return OPTIX_ERROR_INVALID_VALUE;
286
287     OptixResult result;
288
289     OptixStackSizes stackSizesRG = {};
290     result                       = optixProgramGroupGetStackSize(programGroupRG, &stackSizesRG, pipeline);
291     if(result != OPTIX_SUCCESS)
292         return result;
```

```
293
294     OptixStackSizes stackSizesMS1 = {};
295     result                        = optixProgramGroupGetStackSize(programGroupMS1, &stackSizesMS1,
pipeline);
296     if(result != OPTIX_SUCCESS)
297         return result;
298
299     OptixStackSizes stackSizesCH1 = {};
300     for(unsigned int i = 0; i < programGroupCH1Count; ++i)
301     {
302         result = optixUtilAccumulateStackSizes(programGroupCH1[i], &stackSizesCH1, pipeline);
303         if(result != OPTIX_SUCCESS)
304             return result;
305     }
306
307     OptixStackSizes stackSizesMS2 = {};
308     result                        = optixProgramGroupGetStackSize(programGroupMS2, &stackSizesMS2,
pipeline);
309     if(result != OPTIX_SUCCESS)
310         return result;
311
312     OptixStackSizes stackSizesCH2 = {};
313     memset(&stackSizesCH2, 0, sizeof(OptixStackSizes));
314     for(unsigned int i = 0; i < programGroupCH2Count; ++i)
315     {
316         result = optixUtilAccumulateStackSizes(programGroupCH2[i], &stackSizesCH2, pipeline);
317         if(result != OPTIX_SUCCESS)
318             return result;
319     }
320
321     const unsigned int cssRG  = stackSizesRG.cssRG;
322     const unsigned int cssMS1 = stackSizesMS1.cssMS;
323     const unsigned int cssCH1 = stackSizesCH1.cssCH;
324     const unsigned int cssMS2 = stackSizesMS2.cssMS;
325     const unsigned int cssCH2 = stackSizesCH2.cssCH;
326     // no AH, IS, CC, or DC programs
327
328     if(directCallableStackSizeFromTraversal)
329         *directCallableStackSizeFromTraversal = 0;
330     if(directCallableStackSizeFromState)
331         *directCallableStackSizeFromState = 0;
332
333     if(continuationStackSize)
334         *continuationStackSize = cssRG + std::max(cssMS1, cssCH1 + std::max(cssMS2, cssCH2));
335
336     return OPTIX_SUCCESS;
337 }
338   // end group optix_utilities
340
341 #ifdef __cplusplus
342 }
343 #endif
344
345 #endif  // OPTIX_OPTIX_STACK_SIZE_H
```

8.23 optix_stubs.h File Reference

Macros

- #define WIN32_LEAN_AND_MEAN 1

Functions

- static void * optixLoadWindowsDllFromName (const char *optixDllName)
- static void * optixLoadWindowsDll ()

- OPTIXAPI OptixResult optixInitWithHandle (void **handlePtr)
- OPTIXAPI OptixResult optixInit (void)
- OPTIXAPI OptixResult optixUninitWithHandle (void *handle)

Variables

- OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL

8.23.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

8.23.2 Macro Definition Documentation

8.23.2.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN 1
```

8.23.3 Function Documentation

8.23.3.1 optixLoadWindowsDll()

```
static void * optixLoadWindowsDll ( ) [static]
```

8.23.3.2 optixLoadWindowsDllFromName()

```
static void * optixLoadWindowsDllFromName (
          const char * optixDllName ) [static]
```

8.24 optix_stubs.h

Go to the documentation of this file.

```
1 /*
2 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
3 * SPDX-License-Identifier: BSD-3-Clause
4 *
5 * Redistribution and use in source and binary forms, with or without
6 * modification, are permitted provided that the following conditions are met:
7 *
8 * 1. Redistributions of source code must retain the above copyright notice, this
9 * list of conditions and the following disclaimer.
10 *
11 * 2. Redistributions in binary form must reproduce the above copyright notice,
12 * this list of conditions and the following disclaimer in the documentation
13 * and/or other materials provided with the distribution.
14 *
15 * 3. Neither the name of the copyright holder nor the names of its
16 * contributors may be used to endorse or promote products derived from
17 * this software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
20 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
23 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
24 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
```

```
25 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
26 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
27 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
28 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
30
34
35 #ifndef OPTIX_OPTIX_STUBS_H
36 #define OPTIX_OPTIX_STUBS_H
37
38 #include "optix_function_table.h"
39
40 #ifdef _WIN32
41 #ifndef WIN32_LEAN_AND_MEAN
42 #define WIN32_LEAN_AND_MEAN 1
43 #endif
44 #include <windows.h>
45 // The cfgmgr32 header is necessary for interrogating driver information in the registry.
46 // For convenience the library is also linked in automatically using the #pragma command.
47 #include <cfgmgr32.h>
48 #pragma comment(lib, "Cfgmgr32.lib")
49 #include <string.h>
50 #else
51 #include <dlfcn.h>
52 #endif
53
56 #ifndef OPTIXAPI
57 # ifdef OPTIX_ENABLE_SDK_MIXING
58 #   define OPTIXAPI static
59 # else  // OPTIX_ENABLE_SDK_MIXING
60 #   ifdef __cplusplus
61 #     define OPTIXAPI extern "C"
62 #   else  // __cplusplus
63 #     define OPTIXAPI
64 #   endif  // __cplusplus
65 # endif  // OPTIX_ENABLE_SDK_MIXING
66 #endif  // OPTIXAPI
67
68 #ifdef __cplusplus
69 extern "C" {
70 #endif
71
72 // The function table needs to be defined in exactly one translation unit. This can be
73 // achieved by including optix_function_table_definition.h in that translation unit.
74 extern OptixFunctionTable OPTIX_FUNCTION_TABLE_SYMBOL;
75
76 #ifdef __cplusplus
77 }
78 #endif
79
80 #ifdef _WIN32
81 #if defined(_MSC_VER)
82 // Visual Studio produces warnings suggesting strcpy and friends being replaced with _s
83 // variants. All the string lengths and allocation sizes have been calculated and should
84 // be safe, so we are disabling this warning to increase compatibility.
85 #pragma warning(push)
86 #pragma warning(disable : 4996)
87 #endif
88 static void* optixLoadWindowsDllFromName(const char* optixDllName)
89 {
90     void* handle = NULL;
91
92     // Try the bare dll name first.  This picks it up in the local path, followed by
93     //  standard Windows paths.
94     handle = LoadLibraryA((LPSTR)optixDllName);
95     if(handle)
96         return handle;
```

```
97 // If we don't find it in the default dll search path, try the system paths
98
99     // Get the size of the path first, then allocate
100     unsigned int size = GetSystemDirectoryA(NULL, 0);
101     if(size == 0)
102     {
103         // Couldn't get the system path size, so bail
104         return NULL;
105     }
106     size_t pathSize   = size + 1 + strlen(optixDllName);
107     char*  systemPath = (char*)malloc(pathSize);
108     if(systemPath == NULL)
109         return NULL;
110     if(GetSystemDirectoryA(systemPath, size) != size - 1)
111     {
112         // Something went wrong
113         free(systemPath);
114         return NULL;
115     }
116     strcat(systemPath, "\\");
117     strcat(systemPath, optixDllName);
118     handle = LoadLibraryA(systemPath);
119     free(systemPath);
120     if(handle)
121         return handle;
122
123     // If we didn't find it, go looking in the register store.  Since nvoptix.dll doesn't
124     // have its own registry entry, we are going to look for the opengl driver which lives
125     // next to nvoptix.dll.  0 (null) will be returned if any errors occured.
126
127     static const char* deviceInstanceIdentifiersGUID = "{4d36e968-e325-11ce-bfc1-08002be10318}";
128     const ULONG        flags                         = CM_GETIDLIST_FILTER_CLASS |
CM_GETIDLIST_FILTER_PRESENT;
129     ULONG              deviceListSize                = 0;
130     if(CM_Get_Device_ID_List_SizeA(&deviceListSize, deviceInstanceIdentifiersGUID, flags) != CR_SUCCESS)
131     {
132         return NULL;
133     }
134     char* deviceNames = (char*)malloc(deviceListSize);
135     if(deviceNames == NULL)
136         return NULL;
137     if(CM_Get_Device_ID_ListA(deviceInstanceIdentifiersGUID, deviceNames, deviceListSize, flags))
138     {
139         free(deviceNames);
140         return NULL;
141     }
142     DEVINST devID   = 0;
143     char*   dllPath = NULL;
144
145     // Continue to the next device if errors are encountered.
146     for(char* deviceName = deviceNames; *deviceName; deviceName += strlen(deviceName) + 1)
147     {
148         if(CM_Locate_DevNodeA(&devID, deviceName, CM_LOCATE_DEVNODE_NORMAL) != CR_SUCCESS)
149         {
150             continue;
151         }
152         HKEY regKey = 0;
153         if(CM_Open_DevNode_Key(devID, KEY_QUERY_VALUE, 0, RegDisposition_OpenExisting, &regKey,
CM_REGISTRY_SOFTWARE) != CR_SUCCESS)
154         {
155             continue;
156         }
157         const char* valueName = "OpenGLDriverName";
158         DWORD       valueSize = 0;
159         LSTATUS     ret       = RegQueryValueExA(regKey, valueName, NULL, NULL, NULL, &valueSize);
160         if(ret != ERROR_SUCCESS)
161         {
```

```
162             RegCloseKey(regKey);
163             continue;
164         }
165         char* regValue = (char*)malloc(valueSize);
166         if(regValue == NULL)
167         {
168             RegCloseKey(regKey);
169             continue;
170         }
171         ret = RegQueryValueExA(regKey, valueName, NULL, NULL, (LPBYTE)regValue, &valueSize);
172         if(ret != ERROR_SUCCESS)
173         {
174             free(regValue);
175             RegCloseKey(regKey);
176             continue;
177         }
178         // Strip the opengl driver dll name from the string then create a new string with
179         // the path and the nvoptix.dll name
180         for(int i = (int)valueSize - 1; i >= 0 && regValue[i] != '\\'; --i)
181             regValue[i] = '\0';
182         size_t newPathSize = strlen(regValue) + strlen(optixDllName) + 1;
183         dllPath            = (char*)malloc(newPathSize);
184         if(dllPath == NULL)
185         {
186             free(regValue);
187             RegCloseKey(regKey);
188             continue;
189         }
190         strcpy(dllPath, regValue);
191         strcat(dllPath, optixDllName);
192         free(regValue);
193         RegCloseKey(regKey);
194         handle = LoadLibraryA((LPCSTR)dllPath);
195         free(dllPath);
196         if(handle)
197             break;
198     }
199     free(deviceNames);
200     return handle;
201 }
202 #if defined(_MSC_VER)
203 #pragma warning(pop)
204 #endif
205
206 static void* optixLoadWindowsDll()
207 {
208     return optixLoadWindowsDllFromName("nvoptix.dll");
209 }
210 #endif
211
214
224 OPTIXAPI inline OptixResult optixInitWithHandle(void** handlePtr)
225 {
226     // Make sure these functions get initialized to zero in case the DLL and function
227     // table can't be loaded
228     OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorName   = 0;
229     OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorString = 0;
230
231     if(!handlePtr)
232         return OPTIX_ERROR_INVALID_VALUE;
233
234 #ifdef _WIN32
235     *handlePtr = optixLoadWindowsDll();
236     if(!*handlePtr)
237         return OPTIX_ERROR_LIBRARY_NOT_FOUND;
238
239     void* symbol = (void*)GetProcAddress((HMODULE)*handlePtr, "optixQueryFunctionTable");
```

```
240     if(!symbol)
241         return OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND;
242 #else
243     *handlePtr = dlopen("libnvoptix.so.1", RTLD_NOW);
244     if(!*handlePtr)
245         return OPTIX_ERROR_LIBRARY_NOT_FOUND;
246
247     void* symbol = dlsym(*handlePtr, "optixQueryFunctionTable");
248     if(!symbol)
249         return OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND;
250 #endif
251
252     OptixQueryFunctionTable_t* optixQueryFunctionTable = (OptixQueryFunctionTable_t*)symbol;
253
254     return optixQueryFunctionTable(OPTIX_ABI_VERSION, 0, 0, 0, &OPTIX_FUNCTION_TABLE_SYMBOL,
sizeof(OPTIX_FUNCTION_TABLE_SYMBOL));
255 }
256
260 OPTIXAPI inline OptixResult optixInit(void)
261 {
262     void* handle;
263     return optixInitWithHandle(&handle);
264 }
265
271 OPTIXAPI inline OptixResult optixUninitWithHandle(void* handle)
272 {
273     if(!handle)
274         return OPTIX_ERROR_INVALID_VALUE;
275 #ifdef _WIN32
276     if(!FreeLibrary((HMODULE)handle))
277         return OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE;
278 #else
279     if(dlclose(handle))
280         return OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE;
281 #endif
282     OptixFunctionTable empty
283 #ifdef __cplusplus
284       {}
285 #else
286         = { 0 }
287 #endif
288         ;
289     OPTIX_FUNCTION_TABLE_SYMBOL = empty;
290     return OPTIX_SUCCESS;
291 }
292
293   // end group optix_utilities
295
296 #ifndef OPTIX_DOXYGEN_SHOULD_SKIP_THIS
297
298 // Stub functions that forward calls to the corresponding function pointer in the function table.
299
300 OPTIXAPI inline const char* optixGetErrorName(OptixResult result)
301 {
302     if(OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorName)
303         return OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorName(result);
304
305     // If the DLL and symbol table couldn't be loaded, provide a set of error strings
306     // suitable for processing errors related to the DLL loading.
307     switch(result)
308     {
309         case OPTIX_SUCCESS:
310             return "OPTIX_SUCCESS";
311         case OPTIX_ERROR_INVALID_VALUE:
312             return "OPTIX_ERROR_INVALID_VALUE";
313         case OPTIX_ERROR_UNSUPPORTED_ABI_VERSION:
314             return "OPTIX_ERROR_UNSUPPORTED_ABI_VERSION";
```

```
315         case OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH:
316             return "OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH";
317         case OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS:
318             return "OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS";
319         case OPTIX_ERROR_LIBRARY_NOT_FOUND:
320             return "OPTIX_ERROR_LIBRARY_NOT_FOUND";
321         case OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND:
322             return "OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND";
323         case OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE:
324             return "OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE";
325         default:
326             return "Unknown OptixResult code";
327     }
328 }
329
330 OPTIXAPI inline const char* optixGetErrorString(OptixResult result)
331 {
332     if(OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorString)
333         return OPTIX_FUNCTION_TABLE_SYMBOL.optixGetErrorString(result);
334
335     // If the DLL and symbol table couldn't be loaded, provide a set of error strings
336     // suitable for processing errors related to the DLL loading.
337     switch(result)
338     {
339         case OPTIX_SUCCESS:
340             return "Success";
341         case OPTIX_ERROR_INVALID_VALUE:
342             return "Invalid value";
343         case OPTIX_ERROR_UNSUPPORTED_ABI_VERSION:
344             return "Unsupported ABI version";
345         case OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH:
346             return "Function table size mismatch";
347         case OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS:
348             return "Invalid options to entry function";
349         case OPTIX_ERROR_LIBRARY_NOT_FOUND:
350             return "Library not found";
351         case OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND:
352             return "Entry symbol not found";
353         case OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE:
354             return "Library could not be unloaded";
355         default:
356             return "Unknown OptixResult code";
357     }
358 }
359
360 OPTIXAPI inline OptixResult optixDeviceContextCreate(CUcontext fromContext, const
OptixDeviceContextOptions* options, OptixDeviceContext* context)
361 {
362     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextCreate(fromContext, options, context);
363 }
364
365 OPTIXAPI inline OptixResult optixDeviceContextDestroy(OptixDeviceContext context)
366 {
367     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextDestroy(context);
368 }
369
370 OPTIXAPI inline OptixResult optixDeviceContextGetProperty(OptixDeviceContext context,
OptixDeviceProperty property, void* value, size_t sizeInBytes)
371 {
372     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetProperty(context, property, value,
sizeInBytes);
373 }
374
375 OPTIXAPI inline OptixResult optixDeviceContextSetLogCallback(OptixDeviceContext context,
376                                                              OptixLogCallback   callbackFunction,
377                                                              void*              callbackData,
378                                                              unsigned int       callbackLevel)
```

```
379 {
380     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetLogCallback(context, callbackFunction,
callbackData, callbackLevel);
381 }
382
383 OPTIXAPI inline OptixResult optixDeviceContextSetCacheEnabled(OptixDeviceContext context, int enabled)
384 {
385     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetCacheEnabled(context, enabled);
386 }
387
388 OPTIXAPI inline OptixResult optixDeviceContextSetCacheLocation(OptixDeviceContext context, const char*
location)
389 {
390     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetCacheLocation(context, location);
391 }
392
393 OPTIXAPI inline OptixResult optixDeviceContextSetCacheDatabaseSizes(OptixDeviceContext context, size_t
lowWaterMark, size_t highWaterMark)
394 {
395     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextSetCacheDatabaseSizes(context, lowWaterMark,
highWaterMark);
396 }
397
398 OPTIXAPI inline OptixResult optixDeviceContextGetCacheEnabled(OptixDeviceContext context, int* enabled)
399 {
400     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetCacheEnabled(context, enabled);
401 }
402
403 OPTIXAPI inline OptixResult optixDeviceContextGetCacheLocation(OptixDeviceContext context, char*
location, size_t locationSize)
404 {
405     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetCacheLocation(context, location,
locationSize);
406 }
407
408 OPTIXAPI inline OptixResult optixDeviceContextGetCacheDatabaseSizes(OptixDeviceContext context, size_t*
lowWaterMark, size_t* highWaterMark)
409 {
410     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDeviceContextGetCacheDatabaseSizes(context, lowWaterMark,
highWaterMark);
411 }
412
413 OPTIXAPI inline OptixResult optixModuleCreate(OptixDeviceContext                 context,
414                                               const OptixModuleCompileOptions*   moduleCompileOptions,
415                                               const OptixPipelineCompileOptions* pipelineCompileOptions,
416                                               const char*                        input,
417                                               size_t                             inputSize,
418                                               char*                              logString,
419                                               size_t*                            logStringSize,
420                                               OptixModule*                       module)
421 {
422     return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleCreate(context, moduleCompileOptions,
pipelineCompileOptions, input,
423                                                          inputSize, logString, logStringSize, module);
424 }
425
426 OPTIXAPI inline OptixResult optixModuleCreateWithTasks(OptixDeviceContext                 context,
427                                                        const OptixModuleCompileOptions*
moduleCompileOptions,
428                                                        const OptixPipelineCompileOptions*
pipelineCompileOptions,
429                                                        const char*                        input,
430                                                        size_t                             inputSize,
431                                                        char*                              logString,
432                                                        size_t*                            logStringSize,
433                                                        OptixModule*                       module,
434                                                        OptixTask*                         firstTask)
```

```
435 {
436     return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleCreateWithTasks(context, moduleCompileOptions,
pipelineCompileOptions, input,
437                                                                   inputSize, logString, logStringSize,
module, firstTask);
438 }
439
440 OPTIXAPI inline OptixResult optixModuleGetCompilationState(OptixModule module, OptixModuleCompileState*
state)
441 {
442     return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleGetCompilationState(module, state);
443 }
444
445 OPTIXAPI inline OptixResult optixModuleDestroy(OptixModule module)
446 {
447     return OPTIX_FUNCTION_TABLE_SYMBOL.optixModuleDestroy(module);
448 }
449
450 OPTIXAPI inline OptixResult optixBuiltinISModuleGet(OptixDeviceContext                 context,
451                                                     const OptixModuleCompileOptions*   moduleCompileOptions,
452                                                     const OptixPipelineCompileOptions* pipelineCompileOptions,
453                                                     const OptixBuiltinISOptions*       builtinISOptions,
454                                                     OptixModule*                       builtinModule)
455 {
456     return OPTIX_FUNCTION_TABLE_SYMBOL.optixBuiltinISModuleGet(context, moduleCompileOptions,
pipelineCompileOptions,
457                                                                builtinISOptions, builtinModule);
458 }
459
460 OPTIXAPI inline OptixResult optixTaskExecute(OptixTask     task,
461                                              OptixTask*    additionalTasks,
462                                              unsigned int  maxNumAdditionalTasks,
463                                              unsigned int* numAdditionalTasksCreated)
464 {
465     return OPTIX_FUNCTION_TABLE_SYMBOL.optixTaskExecute(task, additionalTasks, maxNumAdditionalTasks,
numAdditionalTasksCreated);
466 }
467
468 OPTIXAPI inline OptixResult optixProgramGroupCreate(OptixDeviceContext             context,
469                                                     const OptixProgramGroupDesc*    programDescriptions,
470                                                     unsigned int                    numProgramGroups,
471                                                     const OptixProgramGroupOptions* options,
472                                                     char*                           logString,
473                                                     size_t*                         logStringSize,
474                                                     OptixProgramGroup*              programGroups)
475 {
476     return OPTIX_FUNCTION_TABLE_SYMBOL.optixProgramGroupCreate(context, programDescriptions,
numProgramGroups, options,
477                                                                logString, logStringSize, programGroups);
478 }
479
480 OPTIXAPI inline OptixResult optixProgramGroupDestroy(OptixProgramGroup programGroup)
481 {
482     return OPTIX_FUNCTION_TABLE_SYMBOL.optixProgramGroupDestroy(programGroup);
483 }
484
485 OPTIXAPI inline OptixResult optixProgramGroupGetStackSize(OptixProgramGroup programGroup,
OptixStackSizes* stackSizes, OptixPipeline pipeline)
486 {
487     return OPTIX_FUNCTION_TABLE_SYMBOL.optixProgramGroupGetStackSize(programGroup, stackSizes, pipeline);
488 }
489
490 OPTIXAPI inline OptixResult optixPipelineCreate(OptixDeviceContext                 context,
491                                                 const OptixPipelineCompileOptions* pipelineCompileOptions,
492                                                 const OptixPipelineLinkOptions*    pipelineLinkOptions,
493                                                 const OptixProgramGroup*           programGroups,
494                                                 unsigned int                       numProgramGroups,
```

```
495                                                 char*                           logString,
496                                                 size_t*                         logStringSize,
497                                                 OptixPipeline*                  pipeline)
498 {
499     return OPTIX_FUNCTION_TABLE_SYMBOL.optixPipelineCreate(context, pipelineCompileOptions,
pipelineLinkOptions, programGroups,
500                                                            numProgramGroups, logString, logStringSize,
pipeline);
501 }
502
503 OPTIXAPI inline OptixResult optixPipelineDestroy(OptixPipeline pipeline)
504 {
505     return OPTIX_FUNCTION_TABLE_SYMBOL.optixPipelineDestroy(pipeline);
506 }
507
508 OPTIXAPI inline OptixResult optixPipelineSetStackSize(OptixPipeline pipeline,
509                                                       unsigned int  directCallableStackSizeFromTraversal,
510                                                       unsigned int  directCallableStackSizeFromState,
511                                                       unsigned int  continuationStackSize,
512                                                       unsigned int  maxTraversableGraphDepth)
513 {
514     return OPTIX_FUNCTION_TABLE_SYMBOL.optixPipelineSetStackSize(pipeline,
directCallableStackSizeFromTraversal,
515                                                                  directCallableStackSizeFromState,
516                                                                  continuationStackSize,
maxTraversableGraphDepth);
517 }
518
519 OPTIXAPI inline OptixResult optixAccelComputeMemoryUsage(OptixDeviceContext           context,
520                                                          const OptixAccelBuildOptions* accelOptions,
521                                                          const OptixBuildInput*        buildInputs,
522                                                          unsigned int                  numBuildInputs,
523                                                          OptixAccelBufferSizes*        bufferSizes)
524 {
525     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelComputeMemoryUsage(context, accelOptions, buildInputs,
numBuildInputs, bufferSizes);
526 }
527
528 OPTIXAPI inline OptixResult optixAccelBuild(OptixDeviceContext           context,
529                                             CUstream                     stream,
530                                             const OptixAccelBuildOptions* accelOptions,
531                                             const OptixBuildInput*        buildInputs,
532                                             unsigned int                  numBuildInputs,
533                                             CUdeviceptr                   tempBuffer,
534                                             size_t                        tempBufferSizeInBytes,
535                                             CUdeviceptr                   outputBuffer,
536                                             size_t                        outputBufferSizeInBytes,
537                                             OptixTraversableHandle*       outputHandle,
538                                             const OptixAccelEmitDesc*     emittedProperties,
539                                             unsigned int                  numEmittedProperties)
540 {
541     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelBuild(context, stream, accelOptions, buildInputs,
numBuildInputs, tempBuffer,
542                                                        tempBufferSizeInBytes, outputBuffer,
outputBufferSizeInBytes,
543                                                    outputHandle, emittedProperties, numEmittedProperties);
544 }
545
546
547 OPTIXAPI inline OptixResult optixAccelGetRelocationInfo(OptixDeviceContext context,
OptixTraversableHandle handle, OptixRelocationInfo* info)
548 {
549     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelGetRelocationInfo(context, handle, info);
550 }
551
552
553 OPTIXAPI inline OptixResult optixCheckRelocationCompatibility(OptixDeviceContext context, const
```

```
OptixRelocationInfo* info, int* compatible)
554 {
555     return OPTIX_FUNCTION_TABLE_SYMBOL.optixCheckRelocationCompatibility(context, info, compatible);
556 }
557
558 OPTIXAPI inline OptixResult optixAccelRelocate(OptixDeviceContext         context,
559                                                CUstream                    stream,
560                                                const OptixRelocationInfo* info,
561                                                const OptixRelocateInput*  relocateInputs,
562                                                size_t                     numRelocateInputs,
563                                                CUdeviceptr                targetAccel,
564                                                size_t                     targetAccelSizeInBytes,
565                                                OptixTraversableHandle*    targetHandle)
566 {
567     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelRelocate(context, stream, info, relocateInputs,
numRelocateInputs,
568                                                           targetAccel, targetAccelSizeInBytes, targetHandle);
569 }
570
571 OPTIXAPI inline OptixResult optixAccelCompact(OptixDeviceContext      context,
572                                               CUstream                stream,
573                                               OptixTraversableHandle  inputHandle,
574                                               CUdeviceptr             outputBuffer,
575                                               size_t                  outputBufferSizeInBytes,
576                                               OptixTraversableHandle* outputHandle)
577 {
578     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelCompact(context, stream, inputHandle, outputBuffer,
579                                                          outputBufferSizeInBytes, outputHandle);
580 }
581
582 OPTIXAPI inline OptixResult optixAccelEmitProperty(OptixDeviceContext        context,
583                                                    CUstream                  stream,
584                                                    OptixTraversableHandle    handle,
585                                                    const OptixAccelEmitDesc* emittedProperty)
586 {
587     return OPTIX_FUNCTION_TABLE_SYMBOL.optixAccelEmitProperty(context, stream, handle, emittedProperty);
588 }
589
590 OPTIXAPI inline OptixResult optixConvertPointerToTraversableHandle(OptixDeviceContext      onDevice,
591                                                                    CUdeviceptr             pointer,
592                                                                    OptixTraversableType    traversableType,
593                                                                    OptixTraversableHandle* traversableHandle)
594 {
595     return OPTIX_FUNCTION_TABLE_SYMBOL.optixConvertPointerToTraversableHandle(onDevice, pointer,
traversableType, traversableHandle);
596 }
597
598 OPTIXAPI inline OptixResult optixOpacityMicromapArrayComputeMemoryUsage(OptixDeviceContext context,
599                                                                         const
OptixOpacityMicromapArrayBuildInput* buildInput,
600                                                                         OptixMicromapBufferSizes* bufferSizes)
601 {
602     return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayComputeMemoryUsage(context, buildInput,
bufferSizes);
603 }
604
605 OPTIXAPI inline OptixResult optixOpacityMicromapArrayBuild(OptixDeviceContext
context,
606                                                            CUstream                                  stream,
607                                                            const OptixOpacityMicromapArrayBuildInput*
buildInput,
608                                                            const OptixMicromapBuffers*               buffers)
609 {
610     return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayBuild(context, stream, buildInput,
buffers);
611 }
612
```

```
613 OPTIXAPI inline OptixResult optixOpacityMicromapArrayGetRelocationInfo(OptixDeviceContext   context,
614                                                                        CUdeviceptr         opacityMicromapArray,
615                                                                        OptixRelocationInfo* info)
616 {
617     return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayGetRelocationInfo(context,
opacityMicromapArray, info);
618 }
619
620 OPTIXAPI inline OptixResult optixOpacityMicromapArrayRelocate(OptixDeviceContext         context,
621                                                               CUstream                   stream,
622                                                               const OptixRelocationInfo* info,
623                                                               CUdeviceptr
targetOpacityMicromapArray,
624                                                               size_t targetOpacityMicromapArraySizeInBytes)
625 {
626     return OPTIX_FUNCTION_TABLE_SYMBOL.optixOpacityMicromapArrayRelocate(context, stream, info,
targetOpacityMicromapArray,
627                                                                          targetOpacityMicromapArraySizeInBytes);
628 }
629
630
631 OPTIXAPI inline OptixResult optixClusterAccelComputeMemoryUsage(OptixDeviceContext
context,
632                                                                 OptixClusterAccelBuildMode        buildMode,
633                                                                 const OptixClusterAccelBuildInput* buildInput,
634                                                                 OptixAccelBufferSizes*            bufferSizes)
635 {
636     return OPTIX_FUNCTION_TABLE_SYMBOL.optixClusterAccelComputeMemoryUsage(context, buildMode,
buildInput, bufferSizes);
637 }
638
639 OPTIXAPI inline OptixResult optixClusterAccelBuild(OptixDeviceContext                   context,
640                                                    CUstream                             stream,
641                                                    const OptixClusterAccelBuildModeDesc* buildModeDesc,
642                                                    const OptixClusterAccelBuildInput*    buildInput,
643                                                    CUdeviceptr                          argsArray,
644                                                    CUdeviceptr                          argsCount,
645                                                    unsigned int                         argsStrideInBytes)
646 {
647     return OPTIX_FUNCTION_TABLE_SYMBOL.optixClusterAccelBuild(context, stream, buildModeDesc,
buildInput, argsArray,
648                                                               argsCount, argsStrideInBytes);
649 }
650
651 OPTIXAPI inline OptixResult optixSbtRecordPackHeader(OptixProgramGroup programGroup, void*
sbtRecordHeaderHostPointer)
652 {
653     return OPTIX_FUNCTION_TABLE_SYMBOL.optixSbtRecordPackHeader(programGroup,
sbtRecordHeaderHostPointer);
654 }
655
656 OPTIXAPI inline OptixResult optixLaunch(OptixPipeline                 pipeline,
657                                         CUstream                      stream,
658                                         CUdeviceptr                   pipelineParams,
659                                         size_t                        pipelineParamsSize,
660                                         const OptixShaderBindingTable* sbt,
661                                         unsigned int                  width,
662                                         unsigned int                  height,
663                                         unsigned int                  depth)
664 {
665     return OPTIX_FUNCTION_TABLE_SYMBOL.optixLaunch(pipeline, stream, pipelineParams, pipelineParamsSize,
sbt, width, height, depth);
666 }
667
668 OPTIXAPI inline OptixResult optixCoopVecMatrixConvert(OptixDeviceContext            context,
669                                                       CUstream                      stream,
670                                                       unsigned int                  numNetworks,
```

```
671                                                 const OptixNetworkDescription* inputNetworkDescription,
672                                                       CUdeviceptr                  inputNetworks,
673                                                  size_t                      inputNetworkStrideInBytes,
674                                                 const OptixNetworkDescription* outputNetworkDescription,
675                                                       CUdeviceptr                  outputNetworks,
676                                                  size_t                      outputNetworkStrideInBytes)
677 {
678     return OPTIX_FUNCTION_TABLE_SYMBOL.optixCoopVecMatrixConvert(context, stream, numNetworks,
inputNetworkDescription,
679                                                                  inputNetworks, inputNetworkStrideInBytes,
outputNetworkDescription,
680                                                                outputNetworks, outputNetworkStrideInBytes);
681 }
682
683 OPTIXAPI inline OptixResult optixCoopVecMatrixComputeSize(OptixDeviceContext       context,
684                                                           unsigned int             N,
685                                                           unsigned int             K,
686                                                           OptixCoopVecElemType     elementType,
687                                                           OptixCoopVecMatrixLayout layout,
688                                                         size_t                   rowColumnStrideInBytes,
689                                                           size_t*                  sizeInBytes)
690 {
691
692     return OPTIX_FUNCTION_TABLE_SYMBOL.optixCoopVecMatrixComputeSize(context, N, K, elementType, layout,
693                                                                      rowColumnStrideInBytes, sizeInBytes);
694 }
695 OPTIXAPI inline OptixResult optixDenoiserCreate(OptixDeviceContext          context,
696                                                 OptixDenoiserModelKind      modelKind,
697                                                 const OptixDenoiserOptions* options,
698                                                 OptixDenoiser*              returnHandle)
699 {
700     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserCreate(context, modelKind, options, returnHandle);
701 }
702
703 OPTIXAPI inline OptixResult optixDenoiserCreateWithUserModel(OptixDeviceContext context,
704                                                              const void*        data,
705                                                              size_t             dataSizeInBytes,
706                                                              OptixDenoiser*     returnHandle)
707 {
708     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserCreateWithUserModel(context, data, dataSizeInBytes,
returnHandle);
709 }
710
711 OPTIXAPI inline OptixResult optixDenoiserDestroy(OptixDenoiser handle)
712 {
713     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserDestroy(handle);
714 }
715
716 OPTIXAPI inline OptixResult optixDenoiserComputeMemoryResources(const OptixDenoiser handle,
717                                                                 unsigned int        maximumInputWidth,
718                                                                 unsigned int        maximumInputHeight,
719                                                                 OptixDenoiserSizes* returnSizes)
720 {
721     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserComputeMemoryResources(handle, maximumInputWidth,
maximumInputHeight, returnSizes);
722 }
723
724 OPTIXAPI inline OptixResult optixDenoiserSetup(OptixDenoiser denoiser,
725                                                CUstream      stream,
726                                                unsigned int  inputWidth,
727                                                unsigned int  inputHeight,
728                                                CUdeviceptr   denoiserState,
729                                                size_t        denoiserStateSizeInBytes,
730                                                CUdeviceptr   scratch,
731                                                size_t        scratchSizeInBytes)
732 {
733     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserSetup(denoiser, stream, inputWidth, inputHeight,
```

```
denoiserState,
734                                                 denoiserStateSizeInBytes, scratch,
scratchSizeInBytes);
735 }
736
737 OPTIXAPI inline OptixResult optixDenoiserInvoke(OptixDenoiser                  handle,
738                                                 CUstream                        stream,
739                                                 const OptixDenoiserParams*      params,
740                                                 CUdeviceptr                     denoiserData,
741                                                 size_t                          denoiserDataSize,
742                                                 const OptixDenoiserGuideLayer* guideLayer,
743                                                 const OptixDenoiserLayer*       layers,
744                                                 unsigned int                    numLayers,
745                                                 unsigned int                    inputOffsetX,
746                                                 unsigned int                    inputOffsetY,
747                                                 CUdeviceptr                     scratch,
748                                                 size_t                          scratchSizeInBytes)
749 {
750     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserInvoke(handle, stream, params, denoiserData,
denoiserDataSize,
751                                                            guideLayer, layers, numLayers, inputOffsetX,
inputOffsetY,
752                                                            scratch, scratchSizeInBytes);
753 }
754
755 OPTIXAPI inline OptixResult optixDenoiserComputeIntensity(OptixDenoiser       handle,
756                                                           CUstream            stream,
757                                                           const OptixImage2D* inputImage,
758                                                           CUdeviceptr         outputIntensity,
759                                                           CUdeviceptr         scratch,
760                                                           size_t              scratchSizeInBytes)
761 {
762     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserComputeIntensity(handle, stream, inputImage,
outputIntensity,
763                                                                      scratch, scratchSizeInBytes);
764 }
765
766 OPTIXAPI inline OptixResult optixDenoiserComputeAverageColor(OptixDenoiser       handle,
767                                                              CUstream            stream,
768                                                              const OptixImage2D* inputImage,
769                                                              CUdeviceptr         outputAverageColor,
770                                                              CUdeviceptr         scratch,
771                                                              size_t              scratchSizeInBytes)
772 {
773     return OPTIX_FUNCTION_TABLE_SYMBOL.optixDenoiserComputeAverageColor(handle, stream, inputImage,
outputAverageColor,
774                                                                         scratch, scratchSizeInBytes);
775 }
776
777 #endif  // OPTIX_DOXYGEN_SHOULD_SKIP_THIS
778
779 #endif  // OPTIX_OPTIX_STUBS_H
```

8.25 optix_types.h File Reference

Classes

- struct OptixDeviceContextOptions
- struct OptixOpacityMicromapUsageCount
- struct OptixBuildInputOpacityMicromap
- struct OptixRelocateInputOpacityMicromap
- struct OptixBuildInputTriangleArray
- struct OptixRelocateInputTriangleArray
- struct OptixBuildInputCurveArray

- struct OptixBuildInputSphereArray
- struct OptixAabb
- struct OptixBuildInputCustomPrimitiveArray
- struct OptixBuildInputInstanceArray
- struct OptixRelocateInputInstanceArray
- struct OptixBuildInput
- struct OptixRelocateInput
- struct OptixInstance
- struct OptixOpacityMicromapDesc
- struct OptixOpacityMicromapHistogramEntry
- struct OptixOpacityMicromapArrayBuildInput
- struct OptixMicromapBufferSizes
- struct OptixMicromapBuffers
- struct OptixMotionOptions
- struct OptixAccelBuildOptions
- struct OptixAccelBufferSizes
- struct OptixAccelEmitDesc
- struct OptixRelocationInfo
- struct OptixStaticTransform
- struct OptixMatrixMotionTransform
- struct OptixSRTData
- struct OptixSRTMotionTransform
- struct OptixClusterAccelBuildModeDescImplicitDest
- struct OptixClusterAccelBuildModeDescExplicitDest
- struct OptixClusterAccelBuildModeDescGetSize
- struct OptixClusterAccelBuildInputTriangles
- struct OptixClusterAccelBuildInputGrids
- struct OptixClusterAccelBuildInputClusters
- struct OptixClusterAccelPrimitiveInfo
- struct OptixClusterAccelBuildInputTrianglesArgs
- struct OptixClusterAccelBuildInputGridsArgs
- struct OptixClusterAccelBuildInputTemplatesArgs
- struct OptixClusterAccelBuildInputClustersArgs
- struct OptixClusterAccelBuildInput
- struct OptixClusterAccelBuildModeDesc
- struct OptixImage2D
- struct OptixDenoiserOptions
- struct OptixDenoiserGuideLayer
- struct OptixDenoiserLayer
- struct OptixDenoiserParams
- struct OptixDenoiserSizes
- struct OptixTraverseData
- struct OptixModuleCompileBoundValueEntry
- struct OptixPayloadType
- struct OptixModuleCompileOptions
- struct OptixBuiltinISOptions
- struct OptixProgramGroupSingleModule
- struct OptixProgramGroupHitgroup
- struct OptixProgramGroupCallables
- struct OptixProgramGroupDesc

- struct OptixProgramGroupOptions
- struct OptixPipelineCompileOptions
- struct OptixPipelineLinkOptions
- struct OptixShaderBindingTable
- struct OptixStackSizes
- struct OptixCoopVecMatrixDescription
- struct OptixNetworkDescription

Macros

- #define OPTIX_SBT_RECORD_HEADER_SIZE ((size_t)32)
- #define OPTIX_SBT_RECORD_ALIGNMENT 16ull
- #define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128ull
- #define OPTIX_INSTANCE_BYTE_ALIGNMENT 16ull
- #define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8ull
- #define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16ull
- #define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
- #define OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
- #define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
- #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
- #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT (0)
- #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE (1)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT (2)
- #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE (3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT (-1)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE (-2)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT (-3)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE (-4)
- #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_CLUSTER_SKIP_OPACITY_MICROMAP (-5)
- #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
- #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12

Typedefs

- typedef unsigned long long CUdeviceptr
- typedef struct OptixDeviceContext_t * OptixDeviceContext
- typedef struct OptixModule_t * OptixModule
- typedef struct OptixProgramGroup_t * OptixProgramGroup
- typedef struct OptixPipeline_t * OptixPipeline
- typedef struct OptixDenoiser_t * OptixDenoiser
- typedef struct OptixTask_t * OptixTask
- typedef unsigned long long OptixTraversableHandle
- typedef unsigned int OptixVisibilityMask
- typedef enum OptixResult OptixResult
- typedef enum OptixDeviceProperty OptixDeviceProperty
- typedef void(* OptixLogCallback) (unsigned int level, const char *tag, const char *message, void *cbdata)
- typedef enum OptixDeviceContextValidationMode OptixDeviceContextValidationMode

- typedef struct OptixDeviceContextOptions OptixDeviceContextOptions
- typedef enum OptixDevicePropertyShaderExecutionReorderingFlags OptixDevicePropertyShaderExecutionReorderingFlags
- typedef enum OptixDevicePropertyClusterAccelFlags OptixDevicePropertyClusterAccelFlags
- typedef enum OptixGeometryFlags OptixGeometryFlags
- typedef enum OptixHitKind OptixHitKind
- typedef enum OptixIndicesFormat OptixIndicesFormat
- typedef enum OptixVertexFormat OptixVertexFormat
- typedef enum OptixTransformFormat OptixTransformFormat
- typedef enum OptixOpacityMicromapFormat OptixOpacityMicromapFormat
- typedef enum OptixOpacityMicromapArrayIndexingMode OptixOpacityMicromapArrayIndexingMode
- typedef struct OptixOpacityMicromapUsageCount OptixOpacityMicromapUsageCount
- typedef struct OptixBuildInputOpacityMicromap OptixBuildInputOpacityMicromap
- typedef struct OptixRelocateInputOpacityMicromap OptixRelocateInputOpacityMicromap
- typedef struct OptixBuildInputTriangleArray OptixBuildInputTriangleArray
- typedef struct OptixRelocateInputTriangleArray OptixRelocateInputTriangleArray
- typedef enum OptixPrimitiveType OptixPrimitiveType
- typedef enum OptixPrimitiveTypeFlags OptixPrimitiveTypeFlags
- typedef enum OptixCurveEndcapFlags OptixCurveEndcapFlags
- typedef struct OptixBuildInputCurveArray OptixBuildInputCurveArray
- typedef struct OptixBuildInputSphereArray OptixBuildInputSphereArray
- typedef struct OptixAabb OptixAabb
- typedef struct OptixBuildInputCustomPrimitiveArray OptixBuildInputCustomPrimitiveArray
- typedef struct OptixBuildInputInstanceArray OptixBuildInputInstanceArray
- typedef struct OptixRelocateInputInstanceArray OptixRelocateInputInstanceArray
- typedef enum OptixBuildInputType OptixBuildInputType
- typedef struct OptixBuildInput OptixBuildInput
- typedef struct OptixRelocateInput OptixRelocateInput
- typedef enum OptixInstanceFlags OptixInstanceFlags
- typedef struct OptixInstance OptixInstance
- typedef enum OptixBuildFlags OptixBuildFlags
- typedef enum OptixOpacityMicromapFlags OptixOpacityMicromapFlags
- typedef struct OptixOpacityMicromapDesc OptixOpacityMicromapDesc
- typedef struct OptixOpacityMicromapHistogramEntry OptixOpacityMicromapHistogramEntry
- typedef struct OptixOpacityMicromapArrayBuildInput OptixOpacityMicromapArrayBuildInput
- typedef struct OptixMicromapBufferSizes OptixMicromapBufferSizes
- typedef struct OptixMicromapBuffers OptixMicromapBuffers
- typedef enum OptixBuildOperation OptixBuildOperation
- typedef enum OptixMotionFlags OptixMotionFlags
- typedef struct OptixMotionOptions OptixMotionOptions
- typedef struct OptixAccelBuildOptions OptixAccelBuildOptions
- typedef struct OptixAccelBufferSizes OptixAccelBufferSizes
- typedef enum OptixAccelPropertyType OptixAccelPropertyType
- typedef struct OptixAccelEmitDesc OptixAccelEmitDesc
- typedef struct OptixRelocationInfo OptixRelocationInfo
- typedef struct OptixStaticTransform OptixStaticTransform
- typedef struct OptixMatrixMotionTransform OptixMatrixMotionTransform
- typedef struct OptixSRTData OptixSRTData
- typedef struct OptixSRTMotionTransform OptixSRTMotionTransform

- typedef enum OptixTraversableType OptixTraversableType
- typedef enum OptixClusterAccelBuildFlags OptixClusterAccelBuildFlags
- typedef enum OptixClusterAccelClusterFlags OptixClusterAccelClusterFlags
- typedef enum OptixClusterAccelPrimitiveFlags OptixClusterAccelPrimitiveFlags
- typedef enum OptixClusterAccelBuildType OptixClusterAccelBuildType
- typedef enum OptixClusterAccelBuildMode OptixClusterAccelBuildMode
- typedef enum OptixClusterAccelIndicesFormat OptixClusterAccelIndicesFormat
- typedef struct OptixClusterAccelBuildModeDescImplicitDest OptixClusterAccelBuildModeDescImplicitDest
- typedef struct OptixClusterAccelBuildModeDescExplicitDest OptixClusterAccelBuildModeDescExplicitDest
- typedef struct OptixClusterAccelBuildModeDescGetSize OptixClusterAccelBuildModeDescGetSize
- typedef struct OptixClusterAccelBuildInputTriangles OptixClusterAccelBuildInputTriangles
- typedef struct OptixClusterAccelBuildInputGrids OptixClusterAccelBuildInputGrids
- typedef struct OptixClusterAccelBuildInputClusters OptixClusterAccelBuildInputClusters
- typedef struct OptixClusterAccelPrimitiveInfo OptixClusterAccelPrimitiveInfo
- typedef enum OptixClusterIDValues OptixClusterIDValues
- typedef struct OptixClusterAccelBuildInputTrianglesArgs OptixClusterAccelBuildInputTrianglesArgs
- typedef struct OptixClusterAccelBuildInputGridsArgs OptixClusterAccelBuildInputGridsArgs
- typedef struct OptixClusterAccelBuildInputTemplatesArgs OptixClusterAccelBuildInputTemplatesArgs
- typedef struct OptixClusterAccelBuildInputClustersArgs OptixClusterAccelBuildInputClustersArgs
- typedef struct OptixClusterAccelBuildInput OptixClusterAccelBuildInput
- typedef struct OptixClusterAccelBuildModeDesc OptixClusterAccelBuildModeDesc
- typedef enum OptixPixelFormat OptixPixelFormat
- typedef struct OptixImage2D OptixImage2D
- typedef enum OptixDenoiserModelKind OptixDenoiserModelKind
- typedef enum OptixDenoiserAlphaMode OptixDenoiserAlphaMode
- typedef struct OptixDenoiserOptions OptixDenoiserOptions
- typedef struct OptixDenoiserGuideLayer OptixDenoiserGuideLayer
- typedef enum OptixDenoiserAOVType OptixDenoiserAOVType
- typedef struct OptixDenoiserLayer OptixDenoiserLayer
- typedef struct OptixDenoiserParams OptixDenoiserParams
- typedef struct OptixDenoiserSizes OptixDenoiserSizes
- typedef enum OptixRayFlags OptixRayFlags
- typedef enum OptixTransformType OptixTransformType
- typedef struct OptixTraverseData OptixTraverseData
- typedef enum OptixTraversableGraphFlags OptixTraversableGraphFlags
- typedef enum OptixCompileOptimizationLevel OptixCompileOptimizationLevel
- typedef enum OptixCompileDebugLevel OptixCompileDebugLevel
- typedef enum OptixModuleCompileState OptixModuleCompileState
- typedef struct OptixModuleCompileBoundValueEntry OptixModuleCompileBoundValueEntry
- typedef enum OptixPayloadTypeID OptixPayloadTypeID
- typedef enum OptixPayloadSemantics OptixPayloadSemantics
- typedef struct OptixPayloadType OptixPayloadType
- typedef struct OptixModuleCompileOptions OptixModuleCompileOptions
- typedef struct OptixBuiltinISOptions OptixBuiltinISOptions

- typedef enum OptixProgramGroupKind OptixProgramGroupKind
- typedef enum OptixProgramGroupFlags OptixProgramGroupFlags
- typedef struct OptixProgramGroupSingleModule OptixProgramGroupSingleModule
- typedef struct OptixProgramGroupHitgroup OptixProgramGroupHitgroup
- typedef struct OptixProgramGroupCallables OptixProgramGroupCallables
- typedef struct OptixProgramGroupDesc OptixProgramGroupDesc
- typedef struct OptixProgramGroupOptions OptixProgramGroupOptions
- typedef enum OptixExceptionCodes OptixExceptionCodes
- typedef enum OptixExceptionFlags OptixExceptionFlags
- typedef struct OptixPipelineCompileOptions OptixPipelineCompileOptions
- typedef struct OptixPipelineLinkOptions OptixPipelineLinkOptions
- typedef struct OptixShaderBindingTable OptixShaderBindingTable
- typedef struct OptixStackSizes OptixStackSizes
- typedef enum OptixDevicePropertyCoopVecFlags OptixDevicePropertyCoopVecFlags
- typedef enum OptixCoopVecElemType OptixCoopVecElemType
- typedef enum OptixCoopVecMatrixLayout OptixCoopVecMatrixLayout
- typedef struct OptixCoopVecMatrixDescription OptixCoopVecMatrixDescription
- typedef struct OptixNetworkDescription OptixNetworkDescription
- typedef enum OptixQueryFunctionTableOptions OptixQueryFunctionTableOptions
- typedef OptixResult() OptixQueryFunctionTable_t(int abiId, unsigned int numOptions, OptixQueryFunctionTableOptions *, const void **, void *functionTable, size_t sizeOfTable)

Enumerations

- enum OptixResult {
  OPTIX_SUCCESS = 0 ,
  OPTIX_ERROR_INVALID_VALUE = 7001 ,
  OPTIX_ERROR_HOST_OUT_OF_MEMORY = 7002 ,
  OPTIX_ERROR_INVALID_OPERATION = 7003 ,
  OPTIX_ERROR_FILE_IO_ERROR = 7004 ,
  OPTIX_ERROR_INVALID_FILE_FORMAT = 7005 ,
  OPTIX_ERROR_DISK_CACHE_INVALID_PATH = 7010 ,
  OPTIX_ERROR_DISK_CACHE_PERMISSION_ERROR = 7011 ,
  OPTIX_ERROR_DISK_CACHE_DATABASE_ERROR = 7012 ,
  OPTIX_ERROR_DISK_CACHE_INVALID_DATA = 7013 ,
  OPTIX_ERROR_LAUNCH_FAILURE = 7050 ,
  OPTIX_ERROR_INVALID_DEVICE_CONTEXT = 7051 ,
  OPTIX_ERROR_CUDA_NOT_INITIALIZED = 7052 ,
  OPTIX_ERROR_VALIDATION_FAILURE = 7053 ,
  OPTIX_ERROR_INVALID_INPUT = 7200 ,
  OPTIX_ERROR_INVALID_LAUNCH_PARAMETER = 7201 ,
  OPTIX_ERROR_INVALID_PAYLOAD_ACCESS = 7202 ,
  OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS = 7203 ,
  OPTIX_ERROR_INVALID_FUNCTION_USE = 7204 ,
  OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS = 7205 ,
  OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250 ,
  OPTIX_ERROR_PIPELINE_LINK_ERROR = 7251 ,
  OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE = 7270 ,
  OPTIX_ERROR_INTERNAL_COMPILER_ERROR = 7299 ,
  OPTIX_ERROR_DENOISER_MODEL_NOT_SET = 7300 ,
  OPTIX_ERROR_DENOISER_NOT_INITIALIZED = 7301 ,
  OPTIX_ERROR_NOT_COMPATIBLE = 7400 ,

OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH = 7500 ,
OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED = 7501 ,
OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID = 7502 ,
OPTIX_ERROR_NOT_SUPPORTED = 7800 ,
OPTIX_ERROR_UNSUPPORTED_ABI_VERSION = 7801 ,
OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH = 7802 ,
OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS = 7803 ,
OPTIX_ERROR_LIBRARY_NOT_FOUND = 7804 ,
OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND = 7805 ,
OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE = 7806 ,
OPTIX_ERROR_DEVICE_OUT_OF_MEMORY = 7807 ,
OPTIX_ERROR_INVALID_POINTER = 7808 ,
OPTIX_ERROR_CUDA_ERROR = 7900 ,
OPTIX_ERROR_INTERNAL_ERROR = 7990 ,
OPTIX_ERROR_UNKNOWN = 7999 }

- enum OptixDeviceProperty {
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004 ,
  OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009 ,
  OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING = 0x200A ,
  OPTIX_DEVICE_PROPERTY_COOP_VEC = 0x200B ,
  OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL = 0x2020 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_VERTICES = 0x2021 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_TRIANGLES = 0x2022 ,
  OPTIX_DEVICE_PROPERTY_LIMIT_MAX_STRUCTURED_GRID_RESOLUTION = 0x2023 }
- 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 OptixDevicePropertyClusterAccelFlags {
  OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_NONE = 0 ,
  OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_STANDARD = 1 << 0 }
- enum OptixGeometryFlags {
  OPTIX_GEOMETRY_FLAG_NONE = 0 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
  OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u << 1 ,
  OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 2 }
- enum OptixHitKind {
  OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE ,
  OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF }
- enum OptixIndicesFormat {
  OPTIX_INDICES_FORMAT_NONE = 0 ,
  OPTIX_INDICES_FORMAT_UNSIGNED_BYTE3 = 0x2101 ,

OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102 ,
OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103 }

- enum OptixVertexFormat {
OPTIX_VERTEX_FORMAT_NONE = 0 ,
OPTIX_VERTEX_FORMAT_FLOAT3 = 0x2121 ,
OPTIX_VERTEX_FORMAT_FLOAT2 = 0x2122 ,
OPTIX_VERTEX_FORMAT_HALF3 = 0x2123 ,
OPTIX_VERTEX_FORMAT_HALF2 = 0x2124 ,
OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125 ,
OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126 }
- enum OptixTransformFormat {
OPTIX_TRANSFORM_FORMAT_NONE = 0 ,
OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1 }
- enum OptixOpacityMicromapFormat {
OPTIX_OPACITY_MICROMAP_FORMAT_NONE = 0 ,
OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE = 1 ,
OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE = 2 }
- enum OptixOpacityMicromapArrayIndexingMode {
OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0 ,
OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1 ,
OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2 }
- enum OptixPrimitiveType {
OPTIX_PRIMITIVE_TYPE_CUSTOM = 0x2500 ,
OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE = 0x2501 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE = 0x2502 ,
OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR = 0x2503 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM = 0x2504 ,
OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE = 0x2505 ,
OPTIX_PRIMITIVE_TYPE_SPHERE = 0x2506 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER = 0x2507 ,
OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE_ROCAPS = 0x2508 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE_ROCAPS = 0x2509 ,
OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM_ROCAPS = 0x250A ,
OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER_ROCAPS = 0x250B ,
OPTIX_PRIMITIVE_TYPE_TRIANGLE = 0x2531 }
- enum OptixPrimitiveTypeFlags {
OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM = 1 << 0 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE = 1 << 1 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE = 1 << 2 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR = 1 << 3 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM = 1 << 4 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE = 1 << 5 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE = 1 << 6 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER = 1 << 7 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE_ROCAPS = 1 << 8 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE_ROCAPS = 1 << 9 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM_ROCAPS = 1 << 10 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER_ROCAPS = 1 << 11 ,
OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE = 1 << 31 }
- enum OptixCurveEndcapFlags {
OPTIX_CURVE_ENDCAP_DEFAULT = 0 ,
OPTIX_CURVE_ENDCAP_ON = 1 << 0 }

- enum OptixBuildInputType {
  OPTIX_BUILD_INPUT_TYPE_TRIANGLES = 0x2141 ,
  OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES = 0x2142 ,
  OPTIX_BUILD_INPUT_TYPE_INSTANCES = 0x2143 ,
  OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS = 0x2144 ,
  OPTIX_BUILD_INPUT_TYPE_CURVES = 0x2145 ,
  OPTIX_BUILD_INPUT_TYPE_SPHERES = 0x2146 }
- enum OptixInstanceFlags {
  OPTIX_INSTANCE_FLAG_NONE = 0 ,
  OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u << 0 ,
  OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u << 1 ,
  OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT = 1u << 2 ,
  OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u << 3 ,
  OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 4 ,
  OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS = 1u << 5 }
- enum OptixBuildFlags {
  OPTIX_BUILD_FLAG_NONE = 0 ,
  OPTIX_BUILD_FLAG_ALLOW_UPDATE = 1u << 0 ,
  OPTIX_BUILD_FLAG_ALLOW_COMPACTION = 1u << 1 ,
  OPTIX_BUILD_FLAG_PREFER_FAST_TRACE = 1u << 2 ,
  OPTIX_BUILD_FLAG_PREFER_FAST_BUILD = 1u << 3 ,
  OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS = 1u << 4 ,
  OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS = 1u << 5 ,
  OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE = 1u << 6 ,
  OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1u << 7 }
- enum OptixOpacityMicromapFlags {
  OPTIX_OPACITY_MICROMAP_FLAG_NONE = 0 ,
  OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 << 0 ,
  OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 << 1 }
- enum OptixBuildOperation {
  OPTIX_BUILD_OPERATION_BUILD = 0x2161 ,
  OPTIX_BUILD_OPERATION_UPDATE = 0x2162 }
- enum OptixMotionFlags {
  OPTIX_MOTION_FLAG_NONE = 0 ,
  OPTIX_MOTION_FLAG_START_VANISH = 1u << 0 ,
  OPTIX_MOTION_FLAG_END_VANISH = 1u << 1 }
- enum OptixAccelPropertyType {
  OPTIX_PROPERTY_TYPE_COMPACTED_SIZE = 0x2181 ,
  OPTIX_PROPERTY_TYPE_AABBS = 0x2182 }
- enum OptixTraversableType {
  OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM = 0x21C1 ,
  OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM = 0x21C2 ,
  OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM = 0x21C3 }
- enum OptixClusterAccelBuildFlags {
  OPTIX_CLUSTER_ACCEL_BUILD_FLAG_NONE = 0 ,
  OPTIX_CLUSTER_ACCEL_BUILD_FLAG_PREFER_FAST_TRACE = 1 << 0 ,
  OPTIX_CLUSTER_ACCEL_BUILD_FLAG_PREFER_FAST_BUILD = 1 << 1 ,
  OPTIX_CLUSTER_ACCEL_BUILD_FLAG_ALLOW_OPACITY_MICROMAPS = 1 << 2 }
- enum OptixClusterAccelClusterFlags {
  OPTIX_CLUSTER_ACCEL_CLUSTER_FLAG_NONE = 0 ,
  OPTIX_CLUSTER_ACCEL_CLUSTER_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1 << 0 }

- enum OptixClusterAccelPrimitiveFlags {
  OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_NONE = 0 ,
  OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1 << 0 ,
  OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1 << 1 ,
  OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_ANYHIT = 1 << 2 }
- enum OptixClusterAccelBuildType {
  OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS = 0x2545 ,
  OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES = 0x2546 ,
  OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES = 0x2547 ,
  OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES = 0x2548 ,
  OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS = 0x2549 }
- enum OptixClusterAccelBuildMode {
  OPTIX_CLUSTER_ACCEL_BUILD_MODE_IMPLICIT_DESTINATIONS = 0 ,
  OPTIX_CLUSTER_ACCEL_BUILD_MODE_EXPLICIT_DESTINATIONS = 1 ,
  OPTIX_CLUSTER_ACCEL_BUILD_MODE_GET_SIZES = 2 }
- enum OptixClusterAccelIndicesFormat {
  OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_8BIT = 1 ,
  OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_16BIT = 2 ,
  OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_32BIT = 4 }
- enum OptixClusterIDValues { OPTIX_CLUSTER_ID_INVALID = 0xFFFFFFFFu }
- enum OptixPixelFormat {
  OPTIX_PIXEL_FORMAT_HALF1 = 0x220a ,
  OPTIX_PIXEL_FORMAT_HALF2 = 0x2207 ,
  OPTIX_PIXEL_FORMAT_HALF3 = 0x2201 ,
  OPTIX_PIXEL_FORMAT_HALF4 = 0x2202 ,
  OPTIX_PIXEL_FORMAT_FLOAT1 = 0x220b ,
  OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208 ,
  OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203 ,
  OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204 ,
  OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205 ,
  OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206 ,
  OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209 }
- enum OptixDenoiserModelKind {
  OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326 ,
  OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328 ,
  OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322 ,
  OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323 ,
  OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325 }
- enum OptixDenoiserAlphaMode {
  OPTIX_DENOISER_ALPHA_MODE_COPY = 0 ,
  OPTIX_DENOISER_ALPHA_MODE_DENOISE = 1 }
- enum OptixDenoiserAOVType {
  OPTIX_DENOISER_AOV_TYPE_NONE = 0 ,
  OPTIX_DENOISER_AOV_TYPE_BEAUTY = 0x7000 ,
  OPTIX_DENOISER_AOV_TYPE_SPECULAR = 0x7001 ,
  OPTIX_DENOISER_AOV_TYPE_REFLECTION = 0x7002 ,
  OPTIX_DENOISER_AOV_TYPE_REFRACTION = 0x7003 ,
  OPTIX_DENOISER_AOV_TYPE_DIFFUSE = 0x7004 }
- enum OptixRayFlags {
  OPTIX_RAY_FLAG_NONE = 0u ,

OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u << 0 ,
OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u << 1 ,
OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u << 2 ,
OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u << 3 ,
OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u << 4 ,
OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u << 5 ,
OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u << 6 ,
OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u << 7 ,
OPTIX_RAY_FLAG_SKIP_TRIANGLES = 1u << 8 ,
OPTIX_RAY_FLAG_SKIP_AABBS = 1u << 9 ,
OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u << 10 }

- enum OptixTransformType {
  OPTIX_TRANSFORM_TYPE_NONE = 0 ,
  OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM = 1 ,
  OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM = 2 ,
  OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM = 3 ,
  OPTIX_TRANSFORM_TYPE_INSTANCE = 4 }
- enum OptixTraversableGraphFlags {
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u << 0 ,
  OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u << 1 }
- enum OptixCompileOptimizationLevel {
  OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342 ,
  OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343 }
- enum OptixCompileDebugLevel {
  OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT = 0 ,
  OPTIX_COMPILE_DEBUG_LEVEL_NONE = 0x2350 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL = 0x2351 ,
  OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353 ,
  OPTIX_COMPILE_DEBUG_LEVEL_FULL = 0x2352 }
- enum OptixModuleCompileState {
  OPTIX_MODULE_COMPILE_STATE_NOT_STARTED = 0x2360 ,
  OPTIX_MODULE_COMPILE_STATE_STARTED = 0x2361 ,
  OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE = 0x2362 ,
  OPTIX_MODULE_COMPILE_STATE_FAILED = 0x2363 ,
  OPTIX_MODULE_COMPILE_STATE_COMPLETED = 0x2364 }
- enum OptixPayloadTypeID {
  OPTIX_PAYLOAD_TYPE_DEFAULT = 0 ,
  OPTIX_PAYLOAD_TYPE_ID_0 = (1 << 0u) ,
  OPTIX_PAYLOAD_TYPE_ID_1 = (1 << 1u) ,
  OPTIX_PAYLOAD_TYPE_ID_2 = (1 << 2u) ,
  OPTIX_PAYLOAD_TYPE_ID_3 = (1 << 3u) ,
  OPTIX_PAYLOAD_TYPE_ID_4 = (1 << 4u) ,
  OPTIX_PAYLOAD_TYPE_ID_5 = (1 << 5u) ,
  OPTIX_PAYLOAD_TYPE_ID_6 = (1 << 6u) ,
  OPTIX_PAYLOAD_TYPE_ID_7 = (1 << 7u) }
- enum OptixPayloadSemantics {
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE = 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ = 1u << 0 ,
  OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE = 2u << 0 ,

OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u << 0 ,
OPTIX_PAYLOAD_SEMANTICS_CH_NONE = 0 ,
OPTIX_PAYLOAD_SEMANTICS_CH_READ = 1u << 2 ,
OPTIX_PAYLOAD_SEMANTICS_CH_WRITE = 2u << 2 ,
OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE = 3u << 2 ,
OPTIX_PAYLOAD_SEMANTICS_MS_NONE = 0 ,
OPTIX_PAYLOAD_SEMANTICS_MS_READ = 1u << 4 ,
OPTIX_PAYLOAD_SEMANTICS_MS_WRITE = 2u << 4 ,
OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE = 3u << 4 ,
OPTIX_PAYLOAD_SEMANTICS_AH_NONE = 0 ,
OPTIX_PAYLOAD_SEMANTICS_AH_READ = 1u << 6 ,
OPTIX_PAYLOAD_SEMANTICS_AH_WRITE = 2u << 6 ,
OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE = 3u << 6 ,
OPTIX_PAYLOAD_SEMANTICS_IS_NONE = 0 ,
OPTIX_PAYLOAD_SEMANTICS_IS_READ = 1u << 8 ,
OPTIX_PAYLOAD_SEMANTICS_IS_WRITE = 2u << 8 ,
OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE = 3u << 8 }

- enum OptixProgramGroupKind {
  OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421 ,
  OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422 ,
  OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423 ,
  OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424 ,
  OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425 }
- enum OptixProgramGroupFlags { OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0 }
- enum OptixExceptionCodes {
  OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1 ,
  OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED = -3 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE = -5 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT = -6 ,
  OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT = -7 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE = -8 ,
  OPTIX_EXCEPTION_CODE_INVALID_RAY = -9 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH = -10 ,
  OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH = -11 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT = -12 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD = -13 ,
  OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD = -14 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS = -15 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0 = -16 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_1 = -17 ,
  OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2 = -18 ,
  OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS = -32 ,
  OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH = -33 }
- enum OptixExceptionFlags {
  OPTIX_EXCEPTION_FLAG_NONE = 0 ,
  OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u << 0 ,
  OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u << 1 ,
  OPTIX_EXCEPTION_FLAG_USER = 1u << 2 ,
  OPTIX_EXCEPTION_FLAG_DEBUG = 1u << 3 }
- enum OptixDevicePropertyCoopVecFlags {
  OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_NONE = 0 ,
  OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_STANDARD = 1 << 0 }

- enum OptixCoopVecElemType {
  OPTIX_COOP_VEC_ELEM_TYPE_UNKNOWN = 0x2A00 ,
  OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16 = 0x2A01 ,
  OPTIX_COOP_VEC_ELEM_TYPE_FLOAT32 = 0x2A03 ,
  OPTIX_COOP_VEC_ELEM_TYPE_UINT8 = 0x2A04 ,
  OPTIX_COOP_VEC_ELEM_TYPE_INT8 = 0x2A05 ,
  OPTIX_COOP_VEC_ELEM_TYPE_UINT32 = 0x2A08 ,
  OPTIX_COOP_VEC_ELEM_TYPE_INT32 = 0x2A09 ,
  OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E4M3 = 0x2A0A ,
  OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E5M2 = 0x2A0B }
- enum OptixCoopVecMatrixLayout {
  OPTIX_COOP_VEC_MATRIX_LAYOUT_ROW_MAJOR = 0x2A40 ,
  OPTIX_COOP_VEC_MATRIX_LAYOUT_COLUMN_MAJOR = 0x2A41 ,
  OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL = 0x2A42 ,
  OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL = 0x2A43 }
- enum OptixQueryFunctionTableOptions { OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0 }

8.25.1 Detailed Description

OptiX public API header.

Author

NVIDIA Corporation

OptiX types include file – defines types and enums used by the API.

8.26 optix_types.h

Go to the documentation of this file.

```
1
2 /*
3 * SPDX-FileCopyrightText: Copyright (c) 2019 - 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.
4 * SPDX-License-Identifier: LicenseRef-NvidiaProprietary
5 *
6 * NVIDIA CORPORATION, its affiliates and licensors retain all intellectual
7 * property and proprietary rights in and to this material, related
8 * documentation and any modifications thereto. Any use, reproduction,
9 * disclosure or distribution of this material and related documentation
10 * without an express license agreement from NVIDIA CORPORATION or
11 * its affiliates is strictly prohibited.
12 */
19
20 #ifndef OPTIX_OPTIX_TYPES_H
21 #define OPTIX_OPTIX_TYPES_H
22
23 #if !defined(__CUDACC_RTC__)
24 #include <stddef.h> /* for size_t */
25 #endif
26
27 #ifdef NV_MODULE_OPTIX
28 // This is a mechanism to include <g_nvconfig.h> in driver builds only and translate any nvconfig macro to
a custom OPTIX-specific macro, that can also be used in SDK builds/installs
29 #include <exp/misc/optix_nvconfig_translate.h>  // includes <g_nvconfig.h>
30 #endif  // NV_MODULE_OPTIX
31
32
35
40 // This typedef should match the one in cuda.h in order to avoid compilation errors.
```

```
41 #if defined(_WIN64) || defined(__LP64__)
43 typedef unsigned long long CUdeviceptr;
44 #else
46 typedef unsigned int CUdeviceptr;
47 #endif
48
50 typedef struct OptixDeviceContext_t* OptixDeviceContext;
51
53 typedef struct OptixModule_t* OptixModule;
54
56 typedef struct OptixProgramGroup_t* OptixProgramGroup;
57
59 typedef struct OptixPipeline_t* OptixPipeline;
60
62 typedef struct OptixDenoiser_t* OptixDenoiser;
63
65 typedef struct OptixTask_t* OptixTask;
66
68 typedef unsigned long long OptixTraversableHandle;
69
71 typedef unsigned int OptixVisibilityMask;
72
74 #define OPTIX_SBT_RECORD_HEADER_SIZE ((size_t)32)
75
77 #define OPTIX_SBT_RECORD_ALIGNMENT 16ull
78
80 #define OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT 128ull
81
83 #define OPTIX_INSTANCE_BYTE_ALIGNMENT 16ull
84
86 #define OPTIX_AABB_BUFFER_BYTE_ALIGNMENT 8ull
87
89 #define OPTIX_GEOMETRY_TRANSFORM_BYTE_ALIGNMENT 16ull
90
92 #define OPTIX_TRANSFORM_BYTE_ALIGNMENT 64ull
93
95 #define OPTIX_OPACITY_MICROMAP_DESC_BUFFER_BYTE_ALIGNMENT 8ull
96
98 #define OPTIX_COMPILE_DEFAULT_MAX_REGISTER_COUNT 0
99
101 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_TYPE_COUNT 8
102
104 #define OPTIX_COMPILE_DEFAULT_MAX_PAYLOAD_VALUE_COUNT 32
105
108 #define OPTIX_OPACITY_MICROMAP_STATE_TRANSPARENT          (0)
109 #define OPTIX_OPACITY_MICROMAP_STATE_OPAQUE               (1)
110 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_TRANSPARENT  (2)
111 #define OPTIX_OPACITY_MICROMAP_STATE_UNKNOWN_OPAQUE       (3)
112
115 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_TRANSPARENT             (-1)
116 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_OPAQUE                  (-2)
117 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_TRANSPARENT     (-3)
118 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_FULLY_UNKNOWN_OPAQUE          (-4)
123 #define OPTIX_OPACITY_MICROMAP_PREDEFINED_INDEX_CLUSTER_SKIP_OPACITY_MICROMAP (-5)
124
126 #define OPTIX_OPACITY_MICROMAP_ARRAY_BUFFER_BYTE_ALIGNMENT 128ull
127
129 #define OPTIX_OPACITY_MICROMAP_MAX_SUBDIVISION_LEVEL 12
130
138 typedef enum OptixResult
139 {
140     OPTIX_SUCCESS                                = 0,
141     OPTIX_ERROR_INVALID_VALUE                    = 7001,
142     OPTIX_ERROR_HOST_OUT_OF_MEMORY               = 7002,
143     OPTIX_ERROR_INVALID_OPERATION                = 7003,
144     OPTIX_ERROR_FILE_IO_ERROR                    = 7004,
145     OPTIX_ERROR_INVALID_FILE_FORMAT              = 7005,
```

```
146     OPTIX_ERROR_DISK_CACHE_INVALID_PATH         = 7010,
147     OPTIX_ERROR_DISK_CACHE_PERMISSION_ERROR     = 7011,
148     OPTIX_ERROR_DISK_CACHE_DATABASE_ERROR       = 7012,
149     OPTIX_ERROR_DISK_CACHE_INVALID_DATA         = 7013,
150     OPTIX_ERROR_LAUNCH_FAILURE                  = 7050,
151     OPTIX_ERROR_INVALID_DEVICE_CONTEXT          = 7051,
152     OPTIX_ERROR_CUDA_NOT_INITIALIZED            = 7052,
153     OPTIX_ERROR_VALIDATION_FAILURE              = 7053,
154     OPTIX_ERROR_INVALID_INPUT                   = 7200,
155     OPTIX_ERROR_INVALID_LAUNCH_PARAMETER        = 7201,
156     OPTIX_ERROR_INVALID_PAYLOAD_ACCESS          = 7202,
157     OPTIX_ERROR_INVALID_ATTRIBUTE_ACCESS        = 7203,
158     OPTIX_ERROR_INVALID_FUNCTION_USE            = 7204,
159     OPTIX_ERROR_INVALID_FUNCTION_ARGUMENTS      = 7205,
160     OPTIX_ERROR_PIPELINE_OUT_OF_CONSTANT_MEMORY = 7250,
161     OPTIX_ERROR_PIPELINE_LINK_ERROR             = 7251,
162     OPTIX_ERROR_ILLEGAL_DURING_TASK_EXECUTE     = 7270,
163     OPTIX_ERROR_INTERNAL_COMPILER_ERROR         = 7299,
164     OPTIX_ERROR_DENOISER_MODEL_NOT_SET          = 7300,
165     OPTIX_ERROR_DENOISER_NOT_INITIALIZED        = 7301,
166     OPTIX_ERROR_NOT_COMPATIBLE                  = 7400,
167     OPTIX_ERROR_PAYLOAD_TYPE_MISMATCH           = 7500,
168     OPTIX_ERROR_PAYLOAD_TYPE_RESOLUTION_FAILED  = 7501,
169     OPTIX_ERROR_PAYLOAD_TYPE_ID_INVALID         = 7502,
170     OPTIX_ERROR_NOT_SUPPORTED                   = 7800,
171     OPTIX_ERROR_UNSUPPORTED_ABI_VERSION         = 7801,
172     OPTIX_ERROR_FUNCTION_TABLE_SIZE_MISMATCH    = 7802,
173     OPTIX_ERROR_INVALID_ENTRY_FUNCTION_OPTIONS  = 7803,
174     OPTIX_ERROR_LIBRARY_NOT_FOUND               = 7804,
175     OPTIX_ERROR_ENTRY_SYMBOL_NOT_FOUND          = 7805,
176     OPTIX_ERROR_LIBRARY_UNLOAD_FAILURE          = 7806,
177     OPTIX_ERROR_DEVICE_OUT_OF_MEMORY            = 7807,
178     OPTIX_ERROR_INVALID_POINTER                 = 7808,
179     OPTIX_ERROR_CUDA_ERROR                      = 7900,
180     OPTIX_ERROR_INTERNAL_ERROR                  = 7990,
181     OPTIX_ERROR_UNKNOWN                         = 7999,
182 } OptixResult;
183
187 typedef enum OptixDeviceProperty
188 {
190     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRACE_DEPTH = 0x2001,
191
194     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_TRAVERSABLE_GRAPH_DEPTH = 0x2002,
195
198     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_PRIMITIVES_PER_GAS = 0x2003,
199
202     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCES_PER_IAS = 0x2004,
203
206     OPTIX_DEVICE_PROPERTY_RTCORE_VERSION = 0x2005,
207
209     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_INSTANCE_ID = 0x2006,
210
213     OPTIX_DEVICE_PROPERTY_LIMIT_NUM_BITS_INSTANCE_VISIBILITY_MASK = 0x2007,
214
217     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_RECORDS_PER_GAS = 0x2008,
218
222     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_SBT_OFFSET = 0x2009,
223
227     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING = 0x200A,
228
232     OPTIX_DEVICE_PROPERTY_COOP_VEC = 0x200B,
233
237     OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL = 0x2020,
238
241     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_VERTICES = 0x2021,
242
245     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_CLUSTER_TRIANGLES = 0x2022,
```

```
246
249     OPTIX_DEVICE_PROPERTY_LIMIT_MAX_STRUCTURED_GRID_RESOLUTION = 0x2023,
250 } OptixDeviceProperty;
251
276 typedef void (*OptixLogCallback)(unsigned int level, const char* tag, const char* message, void* cbdata);
277
291 typedef enum OptixDeviceContextValidationMode
292 {
293     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_OFF = 0,
294     OPTIX_DEVICE_CONTEXT_VALIDATION_MODE_ALL = 0xFFFFFFFF
295 } OptixDeviceContextValidationMode;
296
300 typedef struct OptixDeviceContextOptions
301 {
303     OptixLogCallback logCallbackFunction;
305     void* logCallbackData;
307     int logCallbackLevel;
309     OptixDeviceContextValidationMode validationMode;
310 } OptixDeviceContextOptions;
311
316 typedef enum OptixDevicePropertyShaderExecutionReorderingFlags
317 {
320     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_NONE     = 0,
321
322     // Standard thread reordering is supported
323     OPTIX_DEVICE_PROPERTY_SHADER_EXECUTION_REORDERING_FLAG_STANDARD = 1 « 0,
324 } OptixDevicePropertyShaderExecutionReorderingFlags;
325
330 typedef enum OptixDevicePropertyClusterAccelFlags
331 {
333     OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_NONE     = 0,
334
335     // Cluster acceleration structure builds are supported.
336     OPTIX_DEVICE_PROPERTY_CLUSTER_ACCEL_FLAG_STANDARD = 1 « 0,
337 } OptixDevicePropertyClusterAccelFlags;
338
342 typedef enum OptixGeometryFlags
343 {
345     OPTIX_GEOMETRY_FLAG_NONE = 0,
346
349     OPTIX_GEOMETRY_FLAG_DISABLE_ANYHIT = 1u « 0,
350
354     OPTIX_GEOMETRY_FLAG_REQUIRE_SINGLE_ANYHIT_CALL = 1u « 1,
355
359     OPTIX_GEOMETRY_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u « 2,
360 } OptixGeometryFlags;
361
367 typedef enum OptixHitKind
368 {
370     OPTIX_HIT_KIND_TRIANGLE_FRONT_FACE = 0xFE,
372     OPTIX_HIT_KIND_TRIANGLE_BACK_FACE = 0xFF
373 } OptixHitKind;
374
376 typedef enum OptixIndicesFormat
377 {
379     OPTIX_INDICES_FORMAT_NONE = 0,
381     OPTIX_INDICES_FORMAT_UNSIGNED_BYTE3 = 0x2101,
383     OPTIX_INDICES_FORMAT_UNSIGNED_SHORT3 = 0x2102,
385     OPTIX_INDICES_FORMAT_UNSIGNED_INT3 = 0x2103
386 } OptixIndicesFormat;
387
389 typedef enum OptixVertexFormat
390 {
391     OPTIX_VERTEX_FORMAT_NONE      = 0,
392     OPTIX_VERTEX_FORMAT_FLOAT3    = 0x2121,
393     OPTIX_VERTEX_FORMAT_FLOAT2    = 0x2122,
394     OPTIX_VERTEX_FORMAT_HALF3     = 0x2123,
```

```
395     OPTIX_VERTEX_FORMAT_HALF2     = 0x2124,
396     OPTIX_VERTEX_FORMAT_SNORM16_3 = 0x2125,
397     OPTIX_VERTEX_FORMAT_SNORM16_2 = 0x2126
398 } OptixVertexFormat;
399
401 typedef enum OptixTransformFormat
402 {
403     OPTIX_TRANSFORM_FORMAT_NONE           = 0,
404     OPTIX_TRANSFORM_FORMAT_MATRIX_FLOAT12 = 0x21E1,
405 } OptixTransformFormat;
406
408 typedef enum OptixOpacityMicromapFormat
409 {
411     OPTIX_OPACITY_MICROMAP_FORMAT_NONE = 0,
413     OPTIX_OPACITY_MICROMAP_FORMAT_2_STATE = 1,
415     OPTIX_OPACITY_MICROMAP_FORMAT_4_STATE = 2,
416 } OptixOpacityMicromapFormat;
417
419 typedef enum OptixOpacityMicromapArrayIndexingMode
420 {
422     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_NONE = 0,
425     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_LINEAR = 1,
429     OPTIX_OPACITY_MICROMAP_ARRAY_INDEXING_MODE_INDEXED = 2,
430 } OptixOpacityMicromapArrayIndexingMode;
431
436 typedef struct OptixOpacityMicromapUsageCount
437 {
440     unsigned int count;
442     unsigned int subdivisionLevel;
444     OptixOpacityMicromapFormat format;
445 } OptixOpacityMicromapUsageCount;
446
447 typedef struct OptixBuildInputOpacityMicromap
448 {
450     OptixOpacityMicromapArrayIndexingMode indexingMode;
451
456     CUdeviceptr  opacityMicromapArray;
457
467     CUdeviceptr  indexBuffer;
468
471     unsigned int indexSizeInBytes;
472
475     unsigned int indexStrideInBytes;
476
478     unsigned int indexOffset;
479
481     unsigned int numMicromapUsageCounts;
484     const OptixOpacityMicromapUsageCount* micromapUsageCounts;
485 } OptixBuildInputOpacityMicromap;
486
487 typedef struct OptixRelocateInputOpacityMicromap
488 {
492     CUdeviceptr  opacityMicromapArray;
493 } OptixRelocateInputOpacityMicromap;
494
495
496
500 typedef struct OptixBuildInputTriangleArray
501 {
509     const CUdeviceptr* vertexBuffers;
510
512     unsigned int numVertices;
513
515     OptixVertexFormat vertexFormat;
516
519     unsigned int vertexStrideInBytes;
520
```

```
524     CUdeviceptr indexBuffer;
525
527     unsigned int numIndexTriplets;
528
530     OptixIndicesFormat indexFormat;
531
534     unsigned int indexStrideInBytes;
535
539     CUdeviceptr preTransform;
540
544     const unsigned int* flags;
545
547     unsigned int numSbtRecords;
548
552     CUdeviceptr sbtIndexOffsetBuffer;
553
555     unsigned int sbtIndexOffsetSizeInBytes;
556
559     unsigned int sbtIndexOffsetStrideInBytes;
560
563     unsigned int primitiveIndexOffset;
564
565     // introduced in ABI 24
566     // appended to previous struct
568     OptixTransformFormat transformFormat;
569
571     OptixBuildInputOpacityMicromap opacityMicromap;
572
573 } OptixBuildInputTriangleArray;
574
578 typedef struct OptixRelocateInputTriangleArray
579 {
582     unsigned int numSbtRecords;
583
585     OptixRelocateInputOpacityMicromap opacityMicromap;
586 } OptixRelocateInputTriangleArray;
587
590 typedef enum OptixPrimitiveType
591 {
593     OPTIX_PRIMITIVE_TYPE_CUSTOM                         = 0x2500,
595     OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE        = 0x2501,
597     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE            = 0x2502,
599     OPTIX_PRIMITIVE_TYPE_ROUND_LINEAR                   = 0x2503,
601     OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM               = 0x2504,
603     OPTIX_PRIMITIVE_TYPE_FLAT_QUADRATIC_BSPLINE         = 0x2505,
605     OPTIX_PRIMITIVE_TYPE_SPHERE                         = 0x2506,
607     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER             = 0x2507,
609     OPTIX_PRIMITIVE_TYPE_ROUND_QUADRATIC_BSPLINE_ROCAPS = 0x2508,
611     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BSPLINE_ROCAPS     = 0x2509,
613     OPTIX_PRIMITIVE_TYPE_ROUND_CATMULLROM_ROCAPS        = 0x250A,
615     OPTIX_PRIMITIVE_TYPE_ROUND_CUBIC_BEZIER_ROCAPS      = 0x250B,
616     // OPTIX_PRIMITIVE_TYPE_QUAD                        = 0x2530,
618     OPTIX_PRIMITIVE_TYPE_TRIANGLE                       = 0x2531,
619 } OptixPrimitiveType;
620
624 typedef enum OptixPrimitiveTypeFlags
625 {
627     OPTIX_PRIMITIVE_TYPE_FLAGS_CUSTOM                       = 1 << 0,
629     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE      = 1 << 1,
631     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE          = 1 << 2,
633     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_LINEAR                 = 1 << 3,
635     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM             = 1 << 4,
637     OPTIX_PRIMITIVE_TYPE_FLAGS_FLAT_QUADRATIC_BSPLINE       = 1 << 5,
639     OPTIX_PRIMITIVE_TYPE_FLAGS_SPHERE                       = 1 << 6,
641     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER           = 1 << 7,
643     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_QUADRATIC_BSPLINE_ROCAPS = 1 << 8,
645     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BSPLINE_ROCAPS   = 1 << 9,
```

```
647     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CATMULLROM_ROCAPS      = 1 « 10,
649     OPTIX_PRIMITIVE_TYPE_FLAGS_ROUND_CUBIC_BEZIER_ROCAPS    = 1 « 11,
651     OPTIX_PRIMITIVE_TYPE_FLAGS_TRIANGLE                     = 1 « 31,
653 } OptixPrimitiveTypeFlags;
654
657 typedef enum OptixCurveEndcapFlags
658 {
660     OPTIX_CURVE_ENDCAP_DEFAULT                          = 0,
662     OPTIX_CURVE_ENDCAP_ON                               = 1 « 0,
663 } OptixCurveEndcapFlags;
664
682 typedef struct OptixBuildInputCurveArray
683 {
686     OptixPrimitiveType curveType;
688     unsigned int numPrimitives;
689
694     const CUdeviceptr* vertexBuffers;
696     unsigned int numVertices;
699     unsigned int vertexStrideInBytes;
700
703     const CUdeviceptr* widthBuffers;
706     unsigned int widthStrideInBytes;
707
709     const CUdeviceptr* normalBuffers;
711     unsigned int normalStrideInBytes;
712
718     CUdeviceptr indexBuffer;
721     unsigned int indexStrideInBytes;
722
725     unsigned int flag;
726
729     unsigned int primitiveIndexOffset;
730
732     unsigned int endcapFlags;
733 } OptixBuildInputCurveArray;
734
747 typedef struct OptixBuildInputSphereArray
748 {
753   const CUdeviceptr* vertexBuffers;
754
757   unsigned int vertexStrideInBytes;
759   unsigned int numVertices;
760
763   const CUdeviceptr* radiusBuffers;
766   unsigned int radiusStrideInBytes;
769   int singleRadius;
770
774   const unsigned int* flags;
775
777   unsigned int numSbtRecords;
781   CUdeviceptr sbtIndexOffsetBuffer;
783   unsigned int sbtIndexOffsetSizeInBytes;
786   unsigned int sbtIndexOffsetStrideInBytes;
787
790   unsigned int primitiveIndexOffset;
791 } OptixBuildInputSphereArray;
792
794 typedef struct OptixAabb
795 {
796     float minX;
797     float minY;
798     float minZ;
799     float maxX;
800     float maxY;
801     float maxZ;
802 } OptixAabb;
803
```

```
807 typedef struct OptixBuildInputCustomPrimitiveArray
808 {
813     const CUdeviceptr* aabbBuffers;
814
817     unsigned int numPrimitives;
818
822     unsigned int strideInBytes;
823
827     const unsigned int* flags;
828
830     unsigned int numSbtRecords;
831
835     CUdeviceptr sbtIndexOffsetBuffer;
836
838     unsigned int sbtIndexOffsetSizeInBytes;
839
842     unsigned int sbtIndexOffsetStrideInBytes;
843
846     unsigned int primitiveIndexOffset;
847 } OptixBuildInputCustomPrimitiveArray;
848
852 typedef struct OptixBuildInputInstanceArray
853 {
861     CUdeviceptr instances;
862
864     unsigned int numInstances;
865     // For future consideration: These fields were removed in OPTIX_ABI 43 which needs
866     // to be taken into account if fields are added in the future.
867     //CUdeviceptr aabbs;
868     //unsigned int numAabbs;
869
870     //The following is added in OPTIX_ABI 54
871
875     unsigned int instanceStride;
876 } OptixBuildInputInstanceArray;
877
881 typedef struct OptixRelocateInputInstanceArray
882 {
885     unsigned int numInstances;
886
892     CUdeviceptr traversableHandles;
893
894 } OptixRelocateInputInstanceArray;
895
899 typedef enum OptixBuildInputType
900 {
902     OPTIX_BUILD_INPUT_TYPE_TRIANGLES = 0x2141,
904     OPTIX_BUILD_INPUT_TYPE_CUSTOM_PRIMITIVES = 0x2142,
906     OPTIX_BUILD_INPUT_TYPE_INSTANCES = 0x2143,
908     OPTIX_BUILD_INPUT_TYPE_INSTANCE_POINTERS = 0x2144,
910     OPTIX_BUILD_INPUT_TYPE_CURVES = 0x2145,
912     OPTIX_BUILD_INPUT_TYPE_SPHERES = 0x2146
913 } OptixBuildInputType;
914
920 typedef struct OptixBuildInput
921 {
923     OptixBuildInputType type;
924
925     union
926     {
928         OptixBuildInputTriangleArray triangleArray;
930         OptixBuildInputCurveArray curveArray;
932         OptixBuildInputSphereArray sphereArray;
934         OptixBuildInputCustomPrimitiveArray customPrimitiveArray;
936         OptixBuildInputInstanceArray instanceArray;
937         // TODO Why is the padding so huge? Why is there padding at all?
938         // Why is there an explicit padding, but not for other types like  OptixProgramGroupDesc?
```

```
939         char pad[1024];
940     };
941 } OptixBuildInput;
942
943 // Some 32-bit tools use this header. This static_assert fails for them because
944 // the default enum size is 4 bytes, rather than 8, under 32-bit compilers.
945 // This #ifndef allows them to disable the static assert.
946
947 // TODO Define a static assert for C/pre-C++-11
948 #if defined(__cplusplus) && __cplusplus >= 201103L
949 static_assert(sizeof(OptixBuildInput) == 8 + 1024, "OptixBuildInput has wrong size");
950 #endif
951
955 typedef struct OptixRelocateInput
956 {
958     OptixBuildInputType type;
959
960     union
961     {
963         OptixRelocateInputInstanceArray instanceArray;
964
966         OptixRelocateInputTriangleArray triangleArray;
967
969     };
970 } OptixRelocateInput;
971
978 typedef enum OptixInstanceFlags
979 {
981     OPTIX_INSTANCE_FLAG_NONE = 0,
982
986     OPTIX_INSTANCE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1u « 0,
987
990     OPTIX_INSTANCE_FLAG_FLIP_TRIANGLE_FACING = 1u « 1,
991
995     OPTIX_INSTANCE_FLAG_DISABLE_ANYHIT = 1u « 2,
996
1001     OPTIX_INSTANCE_FLAG_ENFORCE_ANYHIT = 1u « 3,
1002
1009
1011     OPTIX_INSTANCE_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u « 4,
1015     OPTIX_INSTANCE_FLAG_DISABLE_OPACITY_MICROMAPS = 1u « 5,
1016
1017 } OptixInstanceFlags;
1018
1025 typedef struct OptixInstance
1026 {
1028     float transform[12];
1029
1031     unsigned int instanceId;
1032
1036     unsigned int sbtOffset;
1037
1040     unsigned int visibilityMask;
1041
1043     unsigned int flags;
1044
1046     OptixTraversableHandle traversableHandle;
1047
1049     unsigned int pad[2];
1050 } OptixInstance;
1051
1055 typedef enum OptixBuildFlags
1056 {
1058     OPTIX_BUILD_FLAG_NONE = 0,
1059
1062     OPTIX_BUILD_FLAG_ALLOW_UPDATE = 1u « 0,
1063
```

```
1064     OPTIX_BUILD_FLAG_ALLOW_COMPACTION = 1u << 1,
1065
1067     OPTIX_BUILD_FLAG_PREFER_FAST_TRACE = 1u << 2,
1068
1070     OPTIX_BUILD_FLAG_PREFER_FAST_BUILD = 1u << 3,
1071
1086     OPTIX_BUILD_FLAG_ALLOW_RANDOM_VERTEX_ACCESS = 1u << 4,
1087
1090     OPTIX_BUILD_FLAG_ALLOW_RANDOM_INSTANCE_ACCESS = 1u << 5,
1091
1095     OPTIX_BUILD_FLAG_ALLOW_OPACITY_MICROMAP_UPDATE = 1u << 6,
1096
1100     OPTIX_BUILD_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1u << 7,
1101 } OptixBuildFlags;
1102
1103
1105 typedef enum OptixOpacityMicromapFlags
1106 {
1107     OPTIX_OPACITY_MICROMAP_FLAG_NONE = 0,
1108
1110     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_TRACE = 1 << 0,
1111
1113     OPTIX_OPACITY_MICROMAP_FLAG_PREFER_FAST_BUILD = 1 << 1,
1114 } OptixOpacityMicromapFlags;
1115
1117 typedef struct OptixOpacityMicromapDesc
1118 {
1120     unsigned int  byteOffset;
1122     unsigned short subdivisionLevel;
1124     unsigned short format;
1125 } OptixOpacityMicromapDesc;
1126
1131 typedef struct OptixOpacityMicromapHistogramEntry
1132 {
1134     unsigned int              count;
1136     unsigned int              subdivisionLevel;
1138     OptixOpacityMicromapFormat format;
1139 } OptixOpacityMicromapHistogramEntry;
1140
1142 typedef struct OptixOpacityMicromapArrayBuildInput
1143 {
1145     unsigned int flags;
1146
1148     CUdeviceptr inputBuffer;
1149
1152     CUdeviceptr perMicromapDescBuffer;
1153
1157     unsigned int perMicromapDescStrideInBytes;
1158
1160     unsigned int numMicromapHistogramEntries;
1163     const OptixOpacityMicromapHistogramEntry* micromapHistogramEntries;
1164 } OptixOpacityMicromapArrayBuildInput;
1165
1167 typedef struct OptixMicromapBufferSizes
1168 {
1169     size_t outputSizeInBytes;
1170     size_t tempSizeInBytes;
1171 } OptixMicromapBufferSizes;
1172
1174 typedef struct OptixMicromapBuffers
1175 {
1177     CUdeviceptr output;
1179     size_t outputSizeInBytes;
1181     CUdeviceptr temp;
1183     size_t tempSizeInBytes;
1184 } OptixMicromapBuffers;
1185
```

```
1186
1198 typedef enum OptixBuildOperation
1199 {
1201     OPTIX_BUILD_OPERATION_BUILD = 0x2161,
1203     OPTIX_BUILD_OPERATION_UPDATE = 0x2162,
1204 } OptixBuildOperation;
1205
1209 typedef enum OptixMotionFlags
1210 {
1211     OPTIX_MOTION_FLAG_NONE         = 0,
1212     OPTIX_MOTION_FLAG_START_VANISH = 1u « 0,
1213     OPTIX_MOTION_FLAG_END_VANISH   = 1u « 1
1214 } OptixMotionFlags;
1215
1220 typedef struct OptixMotionOptions
1221 {
1224     unsigned short numKeys;
1225
1227     unsigned short flags;
1228
1230     float timeBegin;
1231
1233     float timeEnd;
1234 } OptixMotionOptions;
1235
1239 typedef struct OptixAccelBuildOptions
1240 {
1242     unsigned int buildFlags;
1243
1250     OptixBuildOperation operation;
1251
1253     OptixMotionOptions motionOptions;
1254 } OptixAccelBuildOptions;
1255
1261 typedef struct OptixAccelBufferSizes
1262 {
1265     size_t outputSizeInBytes;
1266
1269     size_t tempSizeInBytes;
1270
1275     size_t tempUpdateSizeInBytes;
1276 } OptixAccelBufferSizes;
1277
1281 typedef enum OptixAccelPropertyType
1282 {
1284     OPTIX_PROPERTY_TYPE_COMPACTED_SIZE = 0x2181,
1285
1287     OPTIX_PROPERTY_TYPE_AABBS = 0x2182,
1288 } OptixAccelPropertyType;
1289
1293 typedef struct OptixAccelEmitDesc
1294 {
1296     CUdeviceptr result;
1297
1299     OptixAccelPropertyType type;
1300 } OptixAccelEmitDesc;
1301
1306 typedef struct OptixRelocationInfo
1307 {
1309     unsigned long long info[4];
1310 } OptixRelocationInfo;
1311
1320 typedef struct OptixStaticTransform
1321 {
1323     OptixTraversableHandle child;
1324
1326     unsigned int pad[2];
```

```
1327
1329     float transform[12];
1330
1333     float invTransform[12];
1334 } OptixStaticTransform;
1335
1363 typedef struct OptixMatrixMotionTransform
1364 {
1366     OptixTraversableHandle child;
1367
1370     OptixMotionOptions motionOptions;
1371
1373     unsigned int pad[3];
1374
1376     float transform[2][12];
1377 } OptixMatrixMotionTransform;
1378
1386 //      [ sx   a   b  pvx ]
1387 //  S = [  0  sy   c  pvy ]
1388 //      [  0   0  sz  pvz ]
1397 //      [  1  0  0 tx ]
1398 //  T = [  0  1  0 ty ]
1399 //      [  0  0  1 tz ]
1409 typedef struct OptixSRTData
1410 {
1413     float sx, a, b, pvx, sy, c, pvy, sz, pvz, qx, qy, qz, qw, tx, ty, tz;
1415 } OptixSRTData;
1416
1417 // TODO Define a static assert for C/pre-C++-11
1418 #if defined(__cplusplus) && __cplusplus >= 201103L
1419 static_assert(sizeof(OptixSRTData) == 16 * 4, "OptixSRTData has wrong size");
1420 #endif
1421
1449 typedef struct OptixSRTMotionTransform
1450 {
1452     OptixTraversableHandle child;
1453
1456     OptixMotionOptions motionOptions;
1457
1459     unsigned int pad[3];
1460
1462     OptixSRTData srtData[2];
1463 } OptixSRTMotionTransform;
1464
1465 // TODO Define a static assert for C/pre-C++-11
1466 #if defined(__cplusplus) && __cplusplus >= 201103L
1467 static_assert(sizeof(OptixSRTMotionTransform) == 8 + 12 + 12 + 2 * 16 * 4, "OptixSRTMotionTransform has
wrong size");
1468 #endif
1469
1473 typedef enum OptixTraversableType
1474 {
1476     OPTIX_TRAVERSABLE_TYPE_STATIC_TRANSFORM = 0x21C1,
1478     OPTIX_TRAVERSABLE_TYPE_MATRIX_MOTION_TRANSFORM = 0x21C2,
1480     OPTIX_TRAVERSABLE_TYPE_SRT_MOTION_TRANSFORM = 0x21C3,
1481 } OptixTraversableType;
1482
1483
1489
1491 typedef enum OptixClusterAccelBuildFlags
1492 {
1493     OPTIX_CLUSTER_ACCEL_BUILD_FLAG_NONE                    = 0,
1494     OPTIX_CLUSTER_ACCEL_BUILD_FLAG_PREFER_FAST_TRACE       = 1 « 0,
1495     OPTIX_CLUSTER_ACCEL_BUILD_FLAG_PREFER_FAST_BUILD       = 1 « 1,
1496     OPTIX_CLUSTER_ACCEL_BUILD_FLAG_ALLOW_OPACITY_MICROMAPS = 1 « 2
1497 } OptixClusterAccelBuildFlags;
1498
```

```
1500 typedef enum OptixClusterAccelClusterFlags
1501 {
1502     OPTIX_CLUSTER_ACCEL_CLUSTER_FLAG_NONE                              = 0,
1505     OPTIX_CLUSTER_ACCEL_CLUSTER_FLAG_ALLOW_DISABLE_OPACITY_MICROMAPS = 1 « 0,
1506 } OptixClusterAccelClusterFlags;
1507
1508 typedef enum OptixClusterAccelPrimitiveFlags
1509 {
1510     OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_NONE                            = 0,
1511     OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_TRIANGLE_FACE_CULLING = 1 « 0,
1512     OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_REQUIRE_SINGLE_ANYHIT_CALL    = 1 « 1,
1513     OPTIX_CLUSTER_ACCEL_PRIMITIVE_FLAG_DISABLE_ANYHIT                = 1 « 2,
1514 } OptixClusterAccelPrimitiveFlags;
1515
1516 // Build type for the multi indirect cluster build
1517 typedef enum OptixClusterAccelBuildType
1518 {
1519     OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS      = 0x2545,
1520     OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES  = 0x2546,
1521     OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES = 0x2547,
1522     OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES  = 0x2548,
1523     OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS     = 0x2549
1524 } OptixClusterAccelBuildType;
1525
1526 typedef enum OptixClusterAccelBuildMode
1527 {
1528     OPTIX_CLUSTER_ACCEL_BUILD_MODE_IMPLICIT_DESTINATIONS = 0,
1529     OPTIX_CLUSTER_ACCEL_BUILD_MODE_EXPLICIT_DESTINATIONS = 1,
1530     OPTIX_CLUSTER_ACCEL_BUILD_MODE_GET_SIZES             = 2
1531 } OptixClusterAccelBuildMode;
1532
1534 typedef enum OptixClusterAccelIndicesFormat
1535 {
1536     OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_8BIT  = 1,
1537     OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_16BIT = 2,
1538     OPTIX_CLUSTER_ACCEL_INDICES_FORMAT_32BIT = 4,
1539 } OptixClusterAccelIndicesFormat;
1540
1541 typedef struct OptixClusterAccelBuildModeDescImplicitDest
1542 {
1547     CUdeviceptr  outputBuffer;               // 128-byte aligned, see OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT
1548     size_t       outputBufferSizeInBytes;     // size of outputHandlesBuffer is
outputHandlesStrideInBytes * number of inputs specified with either argCount or maxArgCount
1549     CUdeviceptr  tempBuffer;                 // 128-byte aligned, see OPTIX_ACCEL_BUFFER_BYTE_ALIGNMENT
1550     size_t       tempBufferSizeInBytes;
1551
1552     CUdeviceptr  outputHandlesBuffer;         // TraversableHandle for GAS, pointer for cluster and
template outputs
1553     unsigned int outputHandlesStrideInBytes;  // minimum 8, 0->8
1554     CUdeviceptr  outputSizesBuffer;           // optional, uint32 array (4 byte aligned?)
1555     unsigned int outputSizesStrideInBytes;    // minimum 4, 0->4
1556 } OptixClusterAccelBuildModeDescImplicitDest;
1557
1558 typedef struct OptixClusterAccelBuildModeDescExplicitDest
1559 {
1560     CUdeviceptr  tempBuffer;
1561     size_t       tempBufferSizeInBytes;
1562     CUdeviceptr  destAddressesBuffer;         // entries must be aligned according to the output type
1563     unsigned int destAddressesStrideInBytes;  // minimum 8, 0->8
1564
1565     CUdeviceptr  outputHandlesBuffer;         // TraversableHandle for GAS, pointer for cluster and
template outputs, can be the same as destAddresses in which case they will overwrite the input
1566     unsigned int outputHandlesStrideInBytes;  // minimum 8, 0->8
1567     CUdeviceptr  outputSizesBuffer;           // optional, uint32 array
1568     unsigned int outputSizesStrideInBytes;    // minimum 4, 0->4
1569 } OptixClusterAccelBuildModeDescExplicitDest;
1570
```

```
1571 typedef struct OptixClusterAccelBuildModeDescGetSize
1572 {
1573     CUdeviceptr  outputSizesBuffer;         // required, uint32 array
1574     unsigned int outputSizesStrideInBytes;  // minimum 4, 0->4
1575     CUdeviceptr  tempBuffer;
1576     size_t       tempBufferSizeInBytes;
1577 } OptixClusterAccelBuildModeDescGetSize;
1578
1579 typedef struct OptixClusterAccelBuildInputTriangles
1580 {
1581     OptixClusterAccelBuildFlags flags;
1582
1585     unsigned int maxArgCount;
1586     OptixVertexFormat vertexFormat;             // OptixVertexFormat (see documentation for supported
formats)
1587     unsigned int maxSbtIndexValue;              // The highest used sbt index over all clusters;
1588                                                 //  this must include the base sbt offset
(::basePrimitiveInfo), any potential per primitive offset (::primitiveInfoBuffer), as well as a potential
offset at template instantiation (OptixClusterAccelBuildInputTemplatesArgs::sbtIndexOffset)
1589     unsigned int maxUniqueSbtIndexCountPerArg;  // Number of unique SBT indices per cluster. If the
cluster has the same
1590                                                 //  SBT index for all its triangles, this value is 1.
1591
1592     unsigned int maxTriangleCountPerArg;        // upper bound on the number of triangles per Arg
1593     unsigned int maxVertexCountPerArg;          // upper bound on the number of vertices per Arg
1594     unsigned int maxTotalTriangleCount;         // optional, upper bound on the number of triangles over
all Args, maxTriangleCountPerArg * maxArgCount otherwise
1595     unsigned int maxTotalVertexCount;           // optional, upper bound on the number of vertices over
all Args, maxVertexCountPerArg * maxArgCount otherwise
1596     unsigned int minPositionTruncateBitCount;   // lower bound on the number of bits being truncated of
the vertex positions.
1597 } OptixClusterAccelBuildInputTriangles;
1598
1599 typedef struct OptixClusterAccelBuildInputGrids
1600 {
1601     OptixClusterAccelBuildFlags flags;
1602     unsigned int                maxArgCount;  // max number of OptixClusterAccelBuildInputGridsArgs
provided at build time for OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS
1603
1604     OptixVertexFormat vertexFormat;      // OptixVertexFormat (see documentation for supported formats)
1605     unsigned int      maxSbtIndexValue;  // the highest used sbt index over all clusters.
1606                                          //  this must include the base sbt offset (::basePrimitiveInfo),
any potential per primitive offset (::primitiveInfoBuffer), as well as a potential offset at template
instantiation (OptixClusterAccelBuildInputTemplatesArgs::sbtIndexOffset)
1607
1608     // Note: hidden in OptiX, set to 1 during conversion. rtcore will also map 0 -> 1
1609     // number of unique SBT indices per cluster. this is always 1 for grids!
1610     // unsigned int maxUniqueSbtIndexCountPerArg (=1);
1611
1612     unsigned int maxWidth;   // the maximum number of edge segments along the width of the grid
1613     unsigned int maxHeight;  // the maximum number of edge segments along the height of the grid
1614 } OptixClusterAccelBuildInputGrids;
1615
1616 typedef struct OptixClusterAccelBuildInputClusters
1617 {
1618     OptixClusterAccelBuildFlags flags;
1619     unsigned int                maxArgCount;  // max number of OptixClusterAccelBuildInputClustersArgs
provided at build time for OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS
1620
1621     unsigned int                maxTotalClusterCount;
1622     unsigned int                maxClusterCountPerArg;
1623 } OptixClusterAccelBuildInputClusters;
1624
1625 typedef struct OptixClusterAccelPrimitiveInfo
1626 {
1627     unsigned int sbtIndex       : 24;
1628     unsigned int reserved       :  5;
```

```
1629     unsigned int primitiveFlags :  3; // combination of OptixClusterAccelPrimitiveFlags
1630 } OptixClusterAccelPrimitiveInfo;
1631
1633 typedef enum OptixClusterIDValues {
1634     OPTIX_CLUSTER_ID_INVALID = 0xFFFFFFFFu,
1635 } OptixClusterIDValues;
1636
1638 typedef struct OptixClusterAccelBuildInputTrianglesArgs
1639 {
1640     unsigned int clusterId;     // 32-bit user-defined ID, for template creation acts as the
baseClusterId and can be offset at template instantiation (see
OptixClusterAccelBuildInputTemplatesArgs::clusterIdOffset)
1641     unsigned int clusterFlags;  // combination of OptixClusterAccelClusterFlags
1642
1643     // packing the following values into a single 32b value
1644     unsigned int triangleCount              : 9;  // max value 256
1645     unsigned int vertexCount                : 9;  // max value 256
1648     unsigned int positionTruncateBitCount   : 6;
1649     unsigned int indexFormat                : 4;  // one of OptixClusterAccelIndicesFormat, 1, 2, or 4
bits-wide indices
1650     unsigned int opacityMicromapIndexFormat : 4;  // one of OptixClusterAccelIndicesFormat, 1, 2, or 4
bits-wide indices
1651
1652     // Applied to all triangles in cluster. Additional per triangle flags can be specified in
PrimitiveInfoBuffer.
1653     OptixClusterAccelPrimitiveInfo basePrimitiveInfo;
1654
1655     unsigned short indexBufferStrideInBytes;  // 0 -> natural stride on all these
1656     unsigned short vertexBufferStrideInBytes;
1657     unsigned short primitiveInfoBufferStrideInBytes;
1658     unsigned short opacityMicromapIndexBufferStrideInBytes;
1659
1660     CUdeviceptr indexBuffer;
1665     CUdeviceptr vertexBuffer;
1666     CUdeviceptr primitiveInfoBuffer;          // Optional, per primitive array of
OptixClusterAccelPrimitiveInfo
1667     CUdeviceptr opacityMicromapArray;
1668     CUdeviceptr opacityMicromapIndexBuffer;
1669
1673     CUdeviceptr instantiationBoundingBoxLimit;
1674 } OptixClusterAccelBuildInputTrianglesArgs;
1675
1677 typedef struct OptixClusterAccelBuildInputGridsArgs
1678 {
1679     unsigned int baseClusterId;   // 32-bit user-defined ID, serves as a base value for the template and
can be offset at template instantiation (see OptixClusterAccelBuildInputTemplatesArgs::clusterIdOffset)
1680     unsigned int clusterFlags;    // combination of OptixClusterAccelClusterFlags
1681
1682     // Applied to all triangles in cluster
1683     OptixClusterAccelPrimitiveInfo basePrimitiveInfo;
1684
1685     // packing the following values into a single 32b value
1686     unsigned int positionTruncateBitCount :  6;
1687     unsigned int reserved                 : 26;
1688
1689     // packing the following values into a single 32b value
1690     unsigned char  dimensions[2];
1691     unsigned short reserved2;
1692 } OptixClusterAccelBuildInputGridsArgs;
1693
1695 typedef struct OptixClusterAccelBuildInputTemplatesArgs
1696 {
1697     unsigned int clusterIdOffset;  // offset applied to template baseClusterId, effective clusterId =
clusterTemplate.baseClusterId + clusterIdOffset. Either may be 0.
1698
1699     unsigned int sbtIndexOffset;   // offset to base sbtIndex from template creation (which may define a
constant or per-triangle base sbtIndex), final sbt index is also limited to fit into 24b
```

```
1700
1701     CUdeviceptr  clusterTemplate;  // opaque pointer to the template
1702     CUdeviceptr  vertexBuffer;     // the vertex data to use to instantiate the template; vertex order
must match that of template creation. For templates created from grids, see documentation.
1703     unsigned int vertexStrideInBytes;
1704     unsigned int reserved;
1705 } OptixClusterAccelBuildInputTemplatesArgs;
1706
1708 typedef struct OptixClusterAccelBuildInputClustersArgs
1709 {
1710     unsigned int clusterHandlesCount;
1711     unsigned int clusterHandlesBufferStrideInBytes;
1715     CUdeviceptr  clusterHandlesBuffer;
1716 } OptixClusterAccelBuildInputClustersArgs;
1717
1718 typedef struct OptixClusterAccelBuildInput
1719 {
1720     OptixClusterAccelBuildType type;
1721
1722     union
1723     {
1724         OptixClusterAccelBuildInputClusters  clusters;    // used for
OPTIX_CLUSTER_ACCEL_BUILD_TYPE_GASES_FROM_CLUSTERS type builds
1725         OptixClusterAccelBuildInputTriangles triangles;   // used for
OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TRIANGLES,
OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_TRIANGLES,
OPTIX_CLUSTER_ACCEL_BUILD_TYPE_CLUSTERS_FROM_TEMPLATES type builds
1726         OptixClusterAccelBuildInputGrids     grids;       // used for
OPTIX_CLUSTER_ACCEL_BUILD_TYPE_TEMPLATES_FROM_GRIDS type builds
1727     };
1728 } OptixClusterAccelBuildInput;
1729
1730 typedef struct OptixClusterAccelBuildModeDesc
1731 {
1732     OptixClusterAccelBuildMode mode;
1733     union
1734     {
1735         OptixClusterAccelBuildModeDescImplicitDest implicitDest;
1736         OptixClusterAccelBuildModeDescExplicitDest explicitDest;
1737         OptixClusterAccelBuildModeDescGetSize      getSize;
1738     };
1739 } OptixClusterAccelBuildModeDesc;
1740
1741
1747
1748
1752 typedef enum OptixPixelFormat
1753 {
1754     OPTIX_PIXEL_FORMAT_HALF1  = 0x220a,
1755     OPTIX_PIXEL_FORMAT_HALF2  = 0x2207,
1756     OPTIX_PIXEL_FORMAT_HALF3  = 0x2201,
1757     OPTIX_PIXEL_FORMAT_HALF4  = 0x2202,
1758     OPTIX_PIXEL_FORMAT_FLOAT1 = 0x220b,
1759     OPTIX_PIXEL_FORMAT_FLOAT2 = 0x2208,
1760     OPTIX_PIXEL_FORMAT_FLOAT3 = 0x2203,
1761     OPTIX_PIXEL_FORMAT_FLOAT4 = 0x2204,
1762     OPTIX_PIXEL_FORMAT_UCHAR3 = 0x2205,
1763     OPTIX_PIXEL_FORMAT_UCHAR4 = 0x2206,
1764     OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER = 0x2209
1765 } OptixPixelFormat;
1766
1770 typedef struct OptixImage2D
1771 {
1773     CUdeviceptr data;
1775     unsigned int width;
1777     unsigned int height;
1779     unsigned int rowStrideInBytes;
```

```
1784     unsigned int pixelStrideInBytes;
1786     OptixPixelFormat format;
1787 } OptixImage2D;
1788
1792 typedef enum OptixDenoiserModelKind
1793 {
1795     OPTIX_DENOISER_MODEL_KIND_AOV = 0x2324,
1796
1797
1799     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_AOV = 0x2326,
1800
1802     OPTIX_DENOISER_MODEL_KIND_UPSCALE2X = 0x2327,
1803
1805     OPTIX_DENOISER_MODEL_KIND_TEMPORAL_UPSCALE2X = 0x2328,
1806
1809     OPTIX_DENOISER_MODEL_KIND_LDR = 0x2322,
1810     OPTIX_DENOISER_MODEL_KIND_HDR = 0x2323,
1811
1813     OPTIX_DENOISER_MODEL_KIND_TEMPORAL = 0x2325
1814
1815 } OptixDenoiserModelKind;
1816
1820 typedef enum OptixDenoiserAlphaMode
1821 {
1823     OPTIX_DENOISER_ALPHA_MODE_COPY = 0,
1824
1826     OPTIX_DENOISER_ALPHA_MODE_DENOISE = 1
1827 } OptixDenoiserAlphaMode;
1828
1832 typedef struct OptixDenoiserOptions
1833 {
1834     // if nonzero, albedo image must be given in OptixDenoiserGuideLayer
1835     unsigned int guideAlbedo;
1836
1837     // if nonzero, normal image must be given in OptixDenoiserGuideLayer
1838     unsigned int guideNormal;
1839
1841     OptixDenoiserAlphaMode denoiseAlpha;
1842 } OptixDenoiserOptions;
1843
1847 typedef struct OptixDenoiserGuideLayer
1848 {
1849     // image with three components: R, G, B.
1850     OptixImage2D  albedo;
1851
1852     // image with two or three components: X, Y, Z.
1853     // (X, Y) camera space for OPTIX_DENOISER_MODEL_KIND_LDR, OPTIX_DENOISER_MODEL_KIND_HDR models.
1854     // (X, Y, Z) world space, all other models.
1855     OptixImage2D  normal;
1856
1857     // image with two components: X, Y.
1858     // pixel movement from previous to current frame for each pixel in screen space.
1859     OptixImage2D  flow;
1860
1861     // Internal images used in temporal AOV denoising modes,
1862     // pixel format OPTIX_PIXEL_FORMAT_INTERNAL_GUIDE_LAYER.
1863     OptixImage2D  previousOutputInternalGuideLayer;
1864     OptixImage2D  outputInternalGuideLayer;
1865
1866     // image with a single component value that specifies how trustworthy the flow vector at x,y
position in
1867     // OptixDenoiserGuideLayer::flow is. Range 0..1 (low->high trustworthiness).
1868     // Ignored if data pointer in the image is zero.
1869     OptixImage2D flowTrustworthiness;
1870
1871 } OptixDenoiserGuideLayer;
1872
```

```
1875 typedef enum OptixDenoiserAOVType
1876 {
1878     OPTIX_DENOISER_AOV_TYPE_NONE       = 0,
1879
1880     OPTIX_DENOISER_AOV_TYPE_BEAUTY     = 0x7000,
1881     OPTIX_DENOISER_AOV_TYPE_SPECULAR   = 0x7001,
1882     OPTIX_DENOISER_AOV_TYPE_REFLECTION = 0x7002,
1883     OPTIX_DENOISER_AOV_TYPE_REFRACTION = 0x7003,
1884     OPTIX_DENOISER_AOV_TYPE_DIFFUSE    = 0x7004
1885
1886 } OptixDenoiserAOVType;
1887
1891 typedef struct OptixDenoiserLayer
1892 {
1893     // input image (beauty or AOV)
1894     OptixImage2D  input;
1895
1896     // denoised output image from previous frame if temporal model kind selected
1897     OptixImage2D  previousOutput;
1898
1899     // denoised output for given input
1900     OptixImage2D  output;
1901
1902     // Type of AOV, used in temporal AOV modes as a hint to improve image quality.
1903     OptixDenoiserAOVType type;
1904 } OptixDenoiserLayer;
1905
1911
1912 typedef struct OptixDenoiserParams
1913 {
1918     CUdeviceptr  hdrIntensity;
1919
1924     float        blendFactor;
1925
1931     CUdeviceptr  hdrAverageColor;
1932
1937     unsigned int temporalModeUsePreviousLayers;
1938 } OptixDenoiserParams;
1939
1943 typedef struct OptixDenoiserSizes
1944 {
1946     size_t stateSizeInBytes;
1947
1950     size_t withOverlapScratchSizeInBytes;
1951
1954     size_t withoutOverlapScratchSizeInBytes;
1955
1957     unsigned int overlapWindowSizeInPixels;
1958
1961     size_t computeAverageColorSizeInBytes;
1962
1965     size_t computeIntensitySizeInBytes;
1966
1968     size_t internalGuideLayerPixelSizeInBytes;
1969 } OptixDenoiserSizes;
1970
1971
1977
1978
1986 typedef enum OptixRayFlags
1987 {
1989     OPTIX_RAY_FLAG_NONE = 0u,
1990
1995     OPTIX_RAY_FLAG_DISABLE_ANYHIT = 1u « 0,
1996
2001     OPTIX_RAY_FLAG_ENFORCE_ANYHIT = 1u « 1,
2002
```

```
2005     OPTIX_RAY_FLAG_TERMINATE_ON_FIRST_HIT = 1u « 2,
2006
2008     OPTIX_RAY_FLAG_DISABLE_CLOSESTHIT = 1u « 3,
2009
2014     OPTIX_RAY_FLAG_CULL_BACK_FACING_TRIANGLES = 1u « 4,
2015
2020     OPTIX_RAY_FLAG_CULL_FRONT_FACING_TRIANGLES = 1u « 5,
2021
2027     OPTIX_RAY_FLAG_CULL_DISABLED_ANYHIT = 1u « 6,
2028
2034     OPTIX_RAY_FLAG_CULL_ENFORCED_ANYHIT = 1u « 7,
2037
2039     OPTIX_RAY_FLAG_SKIP_TRIANGLES         = 1u « 8,
2041     OPTIX_RAY_FLAG_SKIP_AABBS             = 1u « 9,
2042
2044     OPTIX_RAY_FLAG_FORCE_OPACITY_MICROMAP_2_STATE = 1u « 10,
2045 } OptixRayFlags;
2046
2055 typedef enum OptixTransformType
2056 {
2057     OPTIX_TRANSFORM_TYPE_NONE                     = 0,
2058     OPTIX_TRANSFORM_TYPE_STATIC_TRANSFORM         = 1,
2059     OPTIX_TRANSFORM_TYPE_MATRIX_MOTION_TRANSFORM  = 2,
2060     OPTIX_TRANSFORM_TYPE_SRT_MOTION_TRANSFORM     = 3,
2061     OPTIX_TRANSFORM_TYPE_INSTANCE                 = 4,
2062 } OptixTransformType;
2063
2069 typedef struct OptixTraverseData
2070 {
2071     unsigned int data[20];
2072 } OptixTraverseData;
2073
2076 typedef enum OptixTraversableGraphFlags
2077 {
2080     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_ANY = 0,
2081
2085     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_GAS = 1u « 0,
2086
2091     OPTIX_TRAVERSABLE_GRAPH_FLAG_ALLOW_SINGLE_LEVEL_INSTANCING = 1u « 1,
2092 } OptixTraversableGraphFlags;
2093
2097 typedef enum OptixCompileOptimizationLevel
2098 {
2100     OPTIX_COMPILE_OPTIMIZATION_DEFAULT = 0,
2102     OPTIX_COMPILE_OPTIMIZATION_LEVEL_0 = 0x2340,
2104     OPTIX_COMPILE_OPTIMIZATION_LEVEL_1 = 0x2341,
2106     OPTIX_COMPILE_OPTIMIZATION_LEVEL_2 = 0x2342,
2108     OPTIX_COMPILE_OPTIMIZATION_LEVEL_3 = 0x2343,
2109 } OptixCompileOptimizationLevel;
2110
2114 typedef enum OptixCompileDebugLevel
2115 {
2117     OPTIX_COMPILE_DEBUG_LEVEL_DEFAULT  = 0,
2119     OPTIX_COMPILE_DEBUG_LEVEL_NONE     = 0x2350,
2122     OPTIX_COMPILE_DEBUG_LEVEL_MINIMAL  = 0x2351,
2124     OPTIX_COMPILE_DEBUG_LEVEL_MODERATE = 0x2353,
2126     OPTIX_COMPILE_DEBUG_LEVEL_FULL     = 0x2352,
2127 } OptixCompileDebugLevel;
2128
2132 typedef enum OptixModuleCompileState
2133 {
2135     OPTIX_MODULE_COMPILE_STATE_NOT_STARTED       = 0x2360,
2136
2138     OPTIX_MODULE_COMPILE_STATE_STARTED           = 0x2361,
2139
2141     OPTIX_MODULE_COMPILE_STATE_IMPENDING_FAILURE = 0x2362,
2142
```

```
2144     OPTIX_MODULE_COMPILE_STATE_FAILED            = 0x2363,
2145
2147     OPTIX_MODULE_COMPILE_STATE_COMPLETED         = 0x2364,
2148 } OptixModuleCompileState;
2149
2150
2151
2184 typedef struct OptixModuleCompileBoundValueEntry {
2185     size_t pipelineParamOffsetInBytes;
2186     size_t sizeInBytes;
2187     const void* boundValuePtr;
2188     const char* annotation; // optional string to display, set to 0 if unused.  If unused,
2189                             // OptiX will report the annotation as "No annotation"
2190 } OptixModuleCompileBoundValueEntry;
2191
2193 typedef enum OptixPayloadTypeID {
2194     OPTIX_PAYLOAD_TYPE_DEFAULT = 0,
2195     OPTIX_PAYLOAD_TYPE_ID_0 = (1 « 0u),
2196     OPTIX_PAYLOAD_TYPE_ID_1 = (1 « 1u),
2197     OPTIX_PAYLOAD_TYPE_ID_2 = (1 « 2u),
2198     OPTIX_PAYLOAD_TYPE_ID_3 = (1 « 3u),
2199     OPTIX_PAYLOAD_TYPE_ID_4 = (1 « 4u),
2200     OPTIX_PAYLOAD_TYPE_ID_5 = (1 « 5u),
2201     OPTIX_PAYLOAD_TYPE_ID_6 = (1 « 6u),
2202     OPTIX_PAYLOAD_TYPE_ID_7 = (1 « 7u)
2203 } OptixPayloadTypeID;
2204
2218 typedef enum OptixPayloadSemantics
2219 {
2220     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_NONE       = 0,
2221     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ       = 1u « 0,
2222     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_WRITE      = 2u « 0,
2223     OPTIX_PAYLOAD_SEMANTICS_TRACE_CALLER_READ_WRITE = 3u « 0,
2224
2225     OPTIX_PAYLOAD_SEMANTICS_CH_NONE                 = 0,
2226     OPTIX_PAYLOAD_SEMANTICS_CH_READ                 = 1u « 2,
2227     OPTIX_PAYLOAD_SEMANTICS_CH_WRITE                = 2u « 2,
2228     OPTIX_PAYLOAD_SEMANTICS_CH_READ_WRITE           = 3u « 2,
2229
2230     OPTIX_PAYLOAD_SEMANTICS_MS_NONE                 = 0,
2231     OPTIX_PAYLOAD_SEMANTICS_MS_READ                 = 1u « 4,
2232     OPTIX_PAYLOAD_SEMANTICS_MS_WRITE                = 2u « 4,
2233     OPTIX_PAYLOAD_SEMANTICS_MS_READ_WRITE           = 3u « 4,
2234
2235     OPTIX_PAYLOAD_SEMANTICS_AH_NONE                 = 0,
2236     OPTIX_PAYLOAD_SEMANTICS_AH_READ                 = 1u « 6,
2237     OPTIX_PAYLOAD_SEMANTICS_AH_WRITE                = 2u « 6,
2238     OPTIX_PAYLOAD_SEMANTICS_AH_READ_WRITE           = 3u « 6,
2239
2240     OPTIX_PAYLOAD_SEMANTICS_IS_NONE                 = 0,
2241     OPTIX_PAYLOAD_SEMANTICS_IS_READ                 = 1u « 8,
2242     OPTIX_PAYLOAD_SEMANTICS_IS_WRITE                = 2u « 8,
2243     OPTIX_PAYLOAD_SEMANTICS_IS_READ_WRITE           = 3u « 8,
2244 } OptixPayloadSemantics;
2245
2247 typedef struct OptixPayloadType
2248 {
2250     unsigned int numPayloadValues;
2251
2253     const unsigned int *payloadSemantics;
2254 } OptixPayloadType;
2255
2259 typedef struct OptixModuleCompileOptions
2260 {
2263     int maxRegisterCount;
2264
2266     OptixCompileOptimizationLevel optLevel;
```

```
2267
2269     OptixCompileDebugLevel debugLevel;
2270
2272     const OptixModuleCompileBoundValueEntry* boundValues;
2273
2275     unsigned int numBoundValues;
2276
2279     unsigned int numPayloadTypes;
2280
2282     const OptixPayloadType* payloadTypes;
2283
2284 } OptixModuleCompileOptions;
2285
2290 typedef struct OptixBuiltinISOptions
2291 {
2292     OptixPrimitiveType        builtinISModuleType;
2294     int                       usesMotionBlur;
2296     unsigned int              buildFlags;
2298     unsigned int              curveEndcapFlags;
2300 } OptixBuiltinISOptions;
2301
2303 typedef enum OptixProgramGroupKind
2304 {
2307     OPTIX_PROGRAM_GROUP_KIND_RAYGEN = 0x2421,
2308
2311     OPTIX_PROGRAM_GROUP_KIND_MISS = 0x2422,
2312
2315     OPTIX_PROGRAM_GROUP_KIND_EXCEPTION = 0x2423,
2316
2319     OPTIX_PROGRAM_GROUP_KIND_HITGROUP = 0x2424,
2320
2323     OPTIX_PROGRAM_GROUP_KIND_CALLABLES = 0x2425
2324 } OptixProgramGroupKind;
2325
2327 typedef enum OptixProgramGroupFlags
2328 {
2330     OPTIX_PROGRAM_GROUP_FLAGS_NONE = 0
2331 } OptixProgramGroupFlags;
2332
2339 typedef struct OptixProgramGroupSingleModule
2340 {
2342     OptixModule module;
2344     const char* entryFunctionName;
2345 } OptixProgramGroupSingleModule;
2346
2352 typedef struct OptixProgramGroupHitgroup
2353 {
2355     OptixModule moduleCH;
2357     const char* entryFunctionNameCH;
2359     OptixModule moduleAH;
2361     const char* entryFunctionNameAH;
2363     OptixModule moduleIS;
2365     const char* entryFunctionNameIS;
2366 } OptixProgramGroupHitgroup;
2367
2373 typedef struct OptixProgramGroupCallables
2374 {
2376     OptixModule moduleDC;
2378     const char* entryFunctionNameDC;
2380     OptixModule moduleCC;
2382     const char* entryFunctionNameCC;
2383 } OptixProgramGroupCallables;
2384
2386 typedef struct OptixProgramGroupDesc
2387 {
2389     OptixProgramGroupKind kind;
2390
```

```
2392     unsigned int flags;
2393
2394     union
2395     {
2397         OptixProgramGroupSingleModule raygen;
2399         OptixProgramGroupSingleModule miss;
2401         OptixProgramGroupSingleModule exception;
2403         OptixProgramGroupCallables callables;
2405         OptixProgramGroupHitgroup hitgroup;
2406     };
2407 } OptixProgramGroupDesc;
2408
2412 typedef struct OptixProgramGroupOptions
2413 {
2426     const OptixPayloadType* payloadType;
2427 } OptixProgramGroupOptions;
2428
2433 typedef enum OptixExceptionCodes
2434 {
2437     OPTIX_EXCEPTION_CODE_STACK_OVERFLOW = -1,
2438
2441     OPTIX_EXCEPTION_CODE_TRACE_DEPTH_EXCEEDED = -2,
2442
2447     OPTIX_EXCEPTION_CODE_TRAVERSAL_DEPTH_EXCEEDED = -3,
2448
2454     OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_TRAVERSABLE = -5,
2455
2460     OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_MISS_SBT = -6,
2461
2466     //      sbt-index (See optixGetExceptionInvalidSbtOffset),
2467     //      sbt-instance-offset (See OptixInstance::sbtOffset),
2478     OPTIX_EXCEPTION_CODE_TRAVERSAL_INVALID_HIT_SBT = -7,
2479
2482     OPTIX_EXCEPTION_CODE_UNSUPPORTED_PRIMITIVE_TYPE = -8,
2483
2488     OPTIX_EXCEPTION_CODE_INVALID_RAY = -9,
2489
2495     OPTIX_EXCEPTION_CODE_CALLABLE_PARAMETER_MISMATCH = -10,
2496
2498     OPTIX_EXCEPTION_CODE_BUILTIN_IS_MISMATCH = -11,
2499
2504     OPTIX_EXCEPTION_CODE_CALLABLE_INVALID_SBT = -12,
2505
2508     OPTIX_EXCEPTION_CODE_CALLABLE_NO_DC_SBT_RECORD = -13,
2509
2512     OPTIX_EXCEPTION_CODE_CALLABLE_NO_CC_SBT_RECORD = -14,
2513
2520     OPTIX_EXCEPTION_CODE_UNSUPPORTED_SINGLE_LEVEL_GAS = -15,
2521
2524     OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_0 = -16,
2525     OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_1 = -17,
2526     OPTIX_EXCEPTION_CODE_INVALID_VALUE_ARGUMENT_2 = -18,
2529
2531     OPTIX_EXCEPTION_CODE_UNSUPPORTED_DATA_ACCESS = -32,
2532
2534     OPTIX_EXCEPTION_CODE_PAYLOAD_TYPE_MISMATCH = -33,
2535
2536 } OptixExceptionCodes;
2537
2544 typedef enum OptixExceptionFlags
2545 {
2547     OPTIX_EXCEPTION_FLAG_NONE = 0,
2548
2555     OPTIX_EXCEPTION_FLAG_STACK_OVERFLOW = 1u « 0,
2556
2563     OPTIX_EXCEPTION_FLAG_TRACE_DEPTH = 1u « 1,
2564
```

```
2567     OPTIX_EXCEPTION_FLAG_USER = 1u « 2,
2568
2570     OPTIX_EXCEPTION_FLAG_DEBUG = 1u « 3
2571 } OptixExceptionFlags;
2572
2578 typedef struct OptixPipelineCompileOptions
2579 {
2581     int usesMotionBlur;
2582
2584     unsigned int traversableGraphFlags;
2585
2588     int numPayloadValues;
2589
2592     int numAttributeValues;
2593
2595     unsigned int exceptionFlags;
2596
2600     const char* pipelineLaunchParamsVariableName;
2601
2604     unsigned int usesPrimitiveTypeFlags;
2605
2607     int allowOpacityMicromaps;
2608
2610     int allowClusteredGeometry;
2611 } OptixPipelineCompileOptions;
2612
2616 typedef struct OptixPipelineLinkOptions
2617 {
2620     unsigned int maxTraceDepth;
2621
2622     // For future consideration: This field was removed which needs
2623     // to be taken into account if fields are added in the future.
2625     //OptixCompileDebugLevel debugLevel;
2626     // For future consideration: This field was removed which needs
2627     // to be taken into account if fields are added in the future.
2628     //int overrideUsesMotionBlur;
2629 } OptixPipelineLinkOptions;
2630
2634 typedef struct OptixShaderBindingTable
2635 {
2638     CUdeviceptr raygenRecord;
2639
2642     CUdeviceptr exceptionRecord;
2643
2647     CUdeviceptr  missRecordBase;
2648     unsigned int missRecordStrideInBytes;
2649     unsigned int missRecordCount;
2651
2655     CUdeviceptr  hitgroupRecordBase;
2656     unsigned int hitgroupRecordStrideInBytes;
2657     unsigned int hitgroupRecordCount;
2659
2664     CUdeviceptr  callablesRecordBase;
2665     unsigned int callablesRecordStrideInBytes;
2666     unsigned int callablesRecordCount;
2668
2669 } OptixShaderBindingTable;
2670
2674 typedef struct OptixStackSizes
2675 {
2677     unsigned int cssRG;
2679     unsigned int cssMS;
2681     unsigned int cssCH;
2683     unsigned int cssAH;
2685     unsigned int cssIS;
2687     unsigned int cssCC;
2689     unsigned int dssDC;
```

```
2690
2691 } OptixStackSizes;
2692
2693
2699
2704 typedef enum OptixDevicePropertyCoopVecFlags
2705 {
2708     OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_NONE     = 0,
2709
2710     // Standard cooperative vector features are supported
2711     OPTIX_DEVICE_PROPERTY_COOP_VEC_FLAG_STANDARD = 1 « 0,
2712 } OptixDevicePropertyCoopVecFlags;
2713
2714 typedef enum OptixCoopVecElemType
2715 {
2716     OPTIX_COOP_VEC_ELEM_TYPE_UNKNOWN = 0x2A00,
2718     OPTIX_COOP_VEC_ELEM_TYPE_FLOAT16 = 0x2A01,
2720     OPTIX_COOP_VEC_ELEM_TYPE_FLOAT32 = 0x2A03,
2722     OPTIX_COOP_VEC_ELEM_TYPE_UINT8   = 0x2A04,
2724     OPTIX_COOP_VEC_ELEM_TYPE_INT8    = 0x2A05,
2726     OPTIX_COOP_VEC_ELEM_TYPE_UINT32  = 0x2A08,
2728     OPTIX_COOP_VEC_ELEM_TYPE_INT32   = 0x2A09,
2730     OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E4M3 = 0x2A0A,
2732     OPTIX_COOP_VEC_ELEM_TYPE_FLOAT8_E5M2 = 0x2A0B,
2733 } OptixCoopVecElemType;
2734
2735 typedef enum OptixCoopVecMatrixLayout
2736 {
2737     OPTIX_COOP_VEC_MATRIX_LAYOUT_ROW_MAJOR    = 0x2A40,
2738     OPTIX_COOP_VEC_MATRIX_LAYOUT_COLUMN_MAJOR = 0x2A41,
2739 // https://p4viewer.nvidia.com/get///hw/doc/gpu/ampere/ampere/design/IAS/SM/ISA/opcodes/opHMMA.htm
2740     OPTIX_COOP_VEC_MATRIX_LAYOUT_INFERENCING_OPTIMAL = 0x2A42,
2741     OPTIX_COOP_VEC_MATRIX_LAYOUT_TRAINING_OPTIMAL    = 0x2A43,
2742 } OptixCoopVecMatrixLayout;
2743
2750 typedef struct OptixCoopVecMatrixDescription
2751 {
2752     unsigned int            N;
2753     unsigned int            K;
2754     unsigned int            offsetInBytes;
2755     OptixCoopVecElemType    elementType;
2756     OptixCoopVecMatrixLayout layout;
2757     unsigned int            rowColumnStrideInBytes;
2758     unsigned int            sizeInBytes;
2759 } OptixCoopVecMatrixDescription;
2760
2761 typedef struct OptixNetworkDescription
2762 {
2763     OptixCoopVecMatrixDescription* layers;
2764     unsigned int                   numLayers;
2765 } OptixNetworkDescription;
2766
2772
2773
2775 typedef enum OptixQueryFunctionTableOptions
2776 {
2778     OPTIX_QUERY_FUNCTION_TABLE_OPTION_DUMMY = 0
2779
2780 } OptixQueryFunctionTableOptions;
2781
2783 typedef OptixResult(OptixQueryFunctionTable_t)(int          abiId,
2784                                                unsigned int numOptions,
2785                                                OptixQueryFunctionTableOptions* /*optionKeys*/,
2786                                                const void** /*optionValues*/,
2787                                                void*  functionTable,
2788                                                size_t sizeOfTable);
2789
```

```
2790
2791 #if defined(__CUDACC__)
2796 typedef struct OptixInvalidRayExceptionDetails
2797 {
2798     float3 origin;
2799     float3 direction;
2800     float  tmin;
2801     float  tmax;
2802     float  time;
2803 } OptixInvalidRayExceptionDetails;
2804
2811 typedef struct OptixParameterMismatchExceptionDetails
2812 {
2814     unsigned int expectedParameterCount;
2816     unsigned int passedArgumentCount;
2818     unsigned int sbtIndex;
2820     char*        callableName;
2821 } OptixParameterMismatchExceptionDetails;
2822 #endif
2823
2824
2825   // end group optix_types
2827
2828 #endif  // OPTIX_OPTIX_TYPES_H
```

8.27 main.dox File Reference