Protocol

MTLAccelerationStructureCommandEncoder

An object for encoding commands that build or refit acceleration structures.

iOS 14.0+ | iPadOS 14.0+ | Mac Catalyst 14.0+ | macOS 11.0+ | tvOS 16.0+ | visionOS 1.0+

```
protocol MTLAccelerationStructureCommandEncoder : MTLCommandEncoder
```

Mentioned in

- Understanding the Metal 4 core API
- Sampling GPU data into counter sample buffers

Overview

Don't implement this protocol yourself; instead you call methods on an `MTLCommandBuffer` object to create command encoders. Command encoders are lightweight objects that you re-create every time you need to send commands to the GPU.

Topics

Building an acceleration structure

```
func build(accelerationStructure: any MTLAccelerationStructure,
descriptor: MTLAccelerationStructureDescriptor, scratchBuffer: any
MTLBuffer, scratchBufferOffset: Int)
```

Encodes a command to build a new acceleration structure.

Required

Copying an acceleration structure

```
func copy(sourceAccelerationStructure: any MTLAccelerationStructure,
destinationAccelerationStructure: any MTLAccelerationStructure)
```

Encodes a command to copy the data from one acceleration structure to another.

Required

```
func writeCompactedSize(accelerationStructure: any MTLAcceleration
Structure, buffer: any MTLBuffer, offset: Int)
```

Encodes a command to calculate the compacted size of an acceleration structure.

Required

```
func writeCompactedSize(accelerationStructure: any MTLAcceleration
Structure, buffer: any MTLBuffer, offset: Int, sizeDataType: MTLData
Type)
```

Encodes a command to calculate the compacted size of an acceleration structure, taking into account the size of the output data.

Required

```
func copyAndCompact(sourceAccelerationStructure: any MTLAcceleration
Structure, destinationAccelerationStructure: any MTLAcceleration
Structure)
```

Encodes a command to compact an acceleration structure’s data and copy it into a different acceleration structure.

Required

Refitting an acceleration structure

```
func refit(sourceAccelerationStructure: any MTLAccelerationStructure,
descriptor: MTLAccelerationStructureDescriptor, destinationAcceleration
Structure: (any MTLAccelerationStructure)?, scratchBuffer: (any
MTLBuffer)?, scratchBufferOffset: Int)
```

Updates an acceleration structure with new geometry or instance data.

Required

`func refit(sourceAccelerationStructure: any MTLAccelerationStructure, descriptor: MTLAccelerationStructureDescriptor, destinationAcceleration Structure: (any MTLAccelerationStructure)?, scratchBuffer: (any MTLBuffer)?, scratchBufferOffset: Int, options: MTLAcceleration StructureRefitOptions)`

Updates an acceleration structure with new geometry or instance data, with options that control the refitting process.

Required

Synchronizing command execution for untracked resources

`func updateFence(any MTLFence)`

Encodes a command that instructs the GPU to update a fence, which signals passes waiting on the fence.

Required

`func waitForFence(any MTLFence)`

Encodes a command that instructs the GPU to pause before starting the acceleration structure commands until another pass updates a fence.

Required

Specifying resource usage for argument buffers

`func useHeap(any MTLHeap)`

Makes the resources contained in the specified heap available to the acceleration structure pass.

Required

`func useHeaps([any MTLHeap])`

Makes the resources contained in the specified heaps available to the acceleration structure pass.

`func useResource(any MTLResource, usage: MTLResourceUsage)`

Makes a resource available to the acceleration structure pass.

Required

`func useResources([any MTLResource], usage: MTLResourceUsage)`

Makes multiple resources available to the acceleration structure pass.

`struct MTLResourceUsage`

Options that describe how a graphics or compute function uses an argument buffer's resource.

Sampling acceleration structure execution data

`func sampleCounters(sampleBuffer: any MTLCounterSampleBuffer, sampleIndex: Int, barrier: Bool)`

Encodes a command to sample hardware counters at this point in the acceleration structure pass and store the samples into a counter sample buffer.

Required

Relationships

Inherits From

`MTLCommandEncoder`, `NSObjectProtocol`

See Also

Acceleration structures

Improving ray-tracing data access using per-primitive data

Simplify data access and improve GPU utilization by storing custom primitive data directly in the acceleration structure.

`protocol MTLAccelerationStructure`

A collection of model data for GPU-accelerated intersection of rays with the model.

`class MTL4AccelerationStructureDescriptor`

Base class for Metal 4 acceleration structure descriptors.

`class MTLAccelerationStructureDescriptor`

A base class for classes that define the configuration for a new acceleration structure.

`class MTL4PrimitiveAccelerationStructureDescriptor`

Descriptor for a primitive acceleration structure that directly references geometric shapes, such as triangles and bounding boxes.

`class MTLPrimitiveAccelerationStructureDescriptor`

A description of an acceleration structure that contains geometry primitives.

`class MTL4InstanceAccelerationStructureDescriptor`

Descriptor for an instance acceleration structure.

`class MTLInstanceAccelerationStructureDescriptor`

A description of an acceleration structure that derives from instances of primitive acceleration structures.

`struct MTLAccelerationStructureUsage`

Options that affect how Metal builds an acceleration structure and the behavior of that acceleration structure.

`struct MTLAccelerationStructureRefitOptions`