Protocol

MTLBlitCommandEncoder

An interface you can use to encode GPU commands that copy and modify the underlying memory of various Metal resources.

iOS 8.0+ | iPadOS 8.0+ | Mac Catalyst 13.1+ | macOS 10.11+ | tvOS | visionOS 1.0+

```
protocol MTLBlitCommandEncoder : MTLCommandEncoder
```

Mentioned in

- Copying data into or out of mipmaps
- Understanding the Metal 4 core API
- Converting a GPU's counter data into a readable format
- Sampling GPU data into counter sample buffers
- Transferring data between connected GPUs

Overview

Each GPU driver implements the `MTLBlitCommandEncoder` protocol, an interface you use to encode various commands that copy or manipulate resource data, which include the following:

- Filling buffers with repeating bytes
- Generating mipmaps for textures
- Copying data between buffers
- Copying data between textures
- Copying data between a texture and a buffer

- Managing the contents of indirect command buffers
- Synchronizing buffers, textures, and other resources between the CPU and GPU
- Improving runtime performance for resources by optimizing their memory layout for the GPU or CPU

Apps typically use these commands to move data between a resource that uses private storage to, or from, another resource that uses CPU-accessible storage. Some apps use them to apply image-processing and texture effects, such as blurring or reflections, or to render and work with offscreen image data.

You can create an `MTLBlitCommandEncoder` instance by calling one of an `MTLCommandBuffer` instance's methods, such as `makeBlitCommandEncoder()`. When you finish encoding blit commands, finalize the blit pass into the command buffer by calling the encoder's `endEncoding()` method.

Topics

Filling buffers with data

Initialize the bytes within a Metal buffer.

```
func fill(buffer: any MTLBuffer, range: Range<Int>, value: UInt8)
```

Encodes a command that fills a buffer with a constant value for each byte.

Generating texture mipmaps

Initialize a texture's mipmap levels with the content in its primary layer.

```
func generateMipmaps(for: any MTLTexture)
```

Encodes a command that generates mipmaps for a texture from the base mipmap level up to the highest mipmap level.

Required

Copying buffer data to another buffer

Transfer bytes from one Metal buffer to another, such as from a buffer with private storage to a buffer with shared storage.

```
func copy(from: any MTLBuffer, sourceOffset: Int, to: any MTLBuffer, destinationOffset: Int, size: Int)
```

Encodes a command that copies data from one buffer into another.

Required

Copying texture data to another texture

Transfer bytes from one Metal texture to another, whether it's the entire texture or just a portion of it.

```
func copy(from: any MTLTexture, to: any MTLTexture)
```

Encodes a command that copies data from one texture to another.

Required

```
func copy(from: any MTLTexture, sourceSlice: Int, sourceLevel: Int, to: any MTLTexture, destinationSlice: Int, destinationLevel: Int, sliceCount: Int, levelCount: Int)
```

Encodes a command that copies slices of a texture to another texture's slices.

Required

```
func copy(from: any MTLTexture, sourceSlice: Int, sourceLevel: Int, sourceOrigin: MTLOrigin, sourceSize: MTLSize, to: any MTLTexture, destinationSlice: Int, destinationLevel: Int, destinationOrigin: MTLOrigin)
```

Encodes a command that copies image data from a texture's slice into another slice.

Required

Copying buffer data to a texture

Transfer bytes from a Metal buffer into a Metal texture.

```
func copy(from: any MTLBuffer, sourceOffset: Int, sourceBytesPerRow: Int, sourceBytesPerImage: Int, sourceSize: MTLSize, to: any MTLTexture, destinationSlice: Int, destinationLevel: Int, destinationOrigin: MTLOrigin)
```

Encodes a command to copy image data from a source buffer into a destination texture.

Required

```
func copy(from: any MTLBuffer, sourceOffset: Int, sourceBytesPerRow: Int, sourceBytesPerImage: Int, sourceSize: MTLSize, to: any MTLTexture, destinationSlice: Int, destinationLevel: Int, destinationOrigin: MTLOrigin, options: MTLBlitOption)
```

Encodes a command to copy image data from a source buffer into a destination texture.

Required

Copying texture data to a buffer

Transfer bytes from a Metal texture into a Metal buffer.

```
func copy(from: any MTLTexture, sourceSlice: Int, sourceLevel: Int, sourceOrigin: MTLOrigin, sourceSize: MTLSize, to: any MTLBuffer, destinationOffset: Int, destinationBytesPerRow: Int, destinationBytesPerImage: Int)
```

Encodes a command that copies image data from a texture slice to a buffer.

Required

```
func copy(from: any MTLTexture, sourceSlice: Int, sourceLevel: Int, sourceOrigin: MTLOrigin, sourceSize: MTLSize, to: any MTLBuffer, destinationOffset: Int, destinationBytesPerRow: Int, destinationBytesPerImage: Int, options: MTLBlitOption)
```

Encodes a command that copies image data from a texture slice to a buffer, and provides options for special texture formats.

Required

Working with textures on the GPU

Improve the GPU's access times to a texture by altering the layout of its underlying memory.

```
func optimizeContentsForGPUAccess(texture: any MTLTexture)
```

Encodes a command that improves the performance of the GPU's accesses to a texture.

Required

```
func optimizeContentsForGPUAccess(texture: any MTLTexture, slice: Int, level: Int)
```

Encodes a command that improves the performance of the GPU's accesses to a specific portion of a texture.

Required

Working with textures on the CPU

Improve the CPU's access times to a texture by altering the layout of its underlying memory.

```
func optimizeContentsForCPUAccess(texture: any MTLTexture)
```

Encodes a command that improves the performance of the CPU's accesses to a texture.

Required

```
func optimizeContentsForCPUAccess(texture: any MTLTexture, slice: Int, level: Int)
```

Encodes a command that improves the performance of the CPU's accesses to a specific portion of a texture.

Required

Working with managed resources

Update the CPU's copy of a resource that uses the managed storage mode, including buffers and textures, to match the GPU's copy.

```
func synchronize(resource: any MTLResource)
```

Encodes a command that synchronizes the CPU's copy of a managed resource, such as a buffer or texture, so that it matches the GPU's copy.

Required

```
func synchronize(texture: any MTLTexture, slice: Int, level: Int)
```

Encodes a command that synchronizes a part of the CPU's copy of a texture so that it matches the GPU's copy.

Required

Working with fences

Inform the GPU driver when a blit pass needs to wait for resources to update before proceeding, or when it finishes modifying resources itself.

```
func waitForFence(any MTLFence)
```

Encodes a command that instructs the GPU to wait until a pass updates a fence.

Required

```
func updateFence(any MTLFence)
```

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

Required

Working with indirect command buffers

Alter the commands within a Metal indirect command buffer.

```
func copyIndirectCommandBuffer(any MTLIndirectCommandBuffer, sourceRange: Range<Int>, destination: any MTLIndirectCommandBuffer, destinationIndex: Int)
```

Encodes a command that copies commands from one indirect command buffer into another.

```
func resetCommandsInBuffer(any MTLIndirectCommandBuffer, range: Range<Int>)
```

Encodes a command that resets a range of commands in an indirect command buffer.

```
func optimizeIndirectCommandBuffer(any MTLIndirectCommandBuffer, range: Range<Int>)
```

Encodes a command that can improve the performance of a range of commands within an indirect command buffer.

Working with sample counter buffers

Save a GPU's counter data at runtime and then convert it into a usable data structure.

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

Encodes a command that samples the GPU's hardware counters during a blit pass and stores the data in a counter sample buffer.

Required

```
func resolveCounters(any MTLCounterSampleBuffer, range: Range<Int>, destinationBuffer: any MTLBuffer, destinationOffset: Int)
```

Encodes a command that resolves the data from the samples in a sample counter buffer and stores the results into a buffer.

Working with sparse texture access counters

Retrieve or clear the number of times the GPU accesses specific areas within a sparse texture.

```
func getTextureAccessCounters(any MTLTexture, region: MTLRegion, mipLevel: Int, slice: Int, resetCounters: Bool, countersBuffer: any MTLBuffer, countersBufferOffset: Int)
```

Encodes a command that retrieves a sparse texture's access data for a specific region, mipmap level, and slice.

```
func resetTextureAccessCounters(any MTLTexture, region: MTLRegion, mipLevel: Int, slice: Int)
```

Encodes a command that resets a sparse texture's access data for a specific region, mipmap level, and slice.

Instance Methods

`func copy(from: any MTLTensor, sourceOrigin: MTLTensorExtents, sourceDimensions: MTLTensorExtents, to: any MTLTensor, destinationOrigin: MTLTensorExtents, destinationDimensions: MTLTensorExtents)`

Encodes a command to copy data from a slice of one tensor into a slice of another tensor.

Required

Relationships

Inherits From

`MTLCommandEncoder`, `NSObjectProtocol`

See Also

Encoding a blit pass

`struct MTLBlitOption`

The options that enable behavior for some blit operations.