Protocol

MTL4RenderCommandEncoder

Encodes a render pass into a command buffer, including all its draw calls and configuration.

iOS 26.0+ | iPadOS 26.0+ | Mac Catalyst 26.0+ | macOS 26.0+ | tvOS 26.0+ | visionOS 26.0+

```
protocol MTL4RenderCommandEncoder : MTL4CommandEncoder
```

Mentioned in

Understanding the Metal 4 core API

Topics

Instance Properties

`var tileHeight: Int`

Sets the height of a tile for this render pass.

Required

`var tileWidth: Int`

Sets the width of a tile for this render pass.

Required

Instance Methods

```
func dispatchThreadsPerTile(MTLSize)
```

Encodes a command that invokes a tile shader function from the encoder's current tile render pipeline state.

Required

```
func drawIndexedPrimitives(primitiveType: MTLPrimitiveType, indexCount: Int, indexType: MTLIndexType, indexBuffer: MTLGPUAddress, indexBufferLength: Int)
```

Encodes a draw command that renders an instance of a geometric primitive with indexed vertices.

Required

```
func drawIndexedPrimitives(primitiveType: MTLPrimitiveType, indexCount: Int, indexType: MTLIndexType, indexBuffer: MTLGPUAddress, indexBufferLength: Int, instanceCount: Int)
```

Encodes a draw command that renders multiple instances of a geometric primitive with indexed vertices.

Required

```
func drawIndexedPrimitives(primitiveType: MTLPrimitiveType, indexCount: Int, indexType: MTLIndexType, indexBuffer: MTLGPUAddress, indexBufferLength: Int, instanceCount: Int, baseVertex: Int, baseInstance: Int)
```

Encodes a draw command that renders multiple instances of a geometric primitive with indexed vertices, starting with a custom vertex and instance.

Required

```
func drawIndexedPrimitives(primitiveType: MTLPrimitiveType, indexType: MTLIndexType, indexBuffer: MTLGPUAddress, indexBufferLength: Int, indirectBuffer: MTLGPUAddress)
```

Encodes a draw command that renders multiple instances of a geometric primitive with indexed vertices and indirect arguments.

Required

```
func drawMeshThreadgroups(indirectBuffer: MTLGPUAddress, threadsPerObjectThreadgroup: MTLSize, threadsPerMeshThreadgroup: MTLSize)
```

Encodes a draw command that invokes a mesh shader and, optionally, an object shader with indirect arguments.

Required

```
func drawMeshThreadgroups(threadgroupsPerGrid: MTLSize, threadsPerObjectThreadgroup: MTLSize, threadsPerMeshThreadgroup: MTLSize)
```

Encodes a draw command that invokes a mesh shader and, optionally, an object shader with a grid of threadgroups.

Required

`func drawMeshThreads(threadsPerGrid: MTLSize, threadsPerObjectThreadgroup: MTLSize, threadsPerMeshThreadgroup: MTLSize)`

Encodes a draw command that invokes a mesh shader and, optionally, an object shader with a grid of threads.

Required

`func drawPrimitives(primitiveType: MTLPrimitiveType, indirectBuffer: MTLGPUAddress)`

Encodes a draw command that renders multiple instances of a geometric primitive with indirect arguments.

Required

`func drawPrimitives(primitiveType: MTLPrimitiveType, vertexStart: Int, vertexCount: Int)`

Encodes a draw command that renders an instance of a geometric primitive.

Required

`func drawPrimitives(primitiveType: MTLPrimitiveType, vertexStart: Int, vertexCount: Int, instanceCount: Int)`

Encodes a draw command that renders multiple instances of a geometric primitive.

Required

`func drawPrimitives(primitiveType: MTLPrimitiveType, vertexStart: Int, vertexCount: Int, instanceCount: Int, baseInstance: Int)`

Encodes a draw command that renders multiple instances of a geometric primitive, starting with a custom instance identification number.

Required

`func executeCommands(buffer: any MTLIndirectCommandBuffer, indirectBuffer: MTLGPUAddress)`

Encodes a command that runs an indirect range of commands from an indirect command buffer.

Required

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

Encodes a command that runs a range of commands from an indirect command buffer.

`func setArgumentTable(any MTL4ArgumentTable, stages: MTLRenderStages)`

Associates an argument table with a set of render stages.

Required

`func setBlendColor(red: Float, green: Float, blue: Float, alpha: Float)`

Configures each pixel component value, including alpha, for the render pipeline's constant blend color.

Required

`func setColorAttachmentMap(MTLLogicalToPhysicalColorAttachmentMap?)`

Sets the mapping from logical shader color output to physical render pass color attachments.

Required

`func setColorStoreAction(MTLStoreAction, index: Int)`

Configures the store action for a color attachment.

Required

`func setCullMode(MTLCullMode)`

Controls whether Metal culls front facing primitives, back facing primitives, or culls no primitives at all.

Required

`func setDepthBias(Float, slopeScale: Float, clamp: Float)`

Configures the adjustments a render pass applies to depth values from fragment shader functions by a scaling factor and bias.

Required

`func setDepthClipMode(MTLDepthClipMode)`

Controls the behavior for fragments outside of the near or far planes.

Required

`func setDepthStencilState((any MTLDepthStencilState)?)`

Configures this encoder with a depth stencil state that applies to your subsequent draw commands.

Required

`func setDepthStoreAction(MTLStoreAction)`

Configures the store action for the depth attachment.

Required

`func setDepthTestBounds(ClosedRange<Float>)`

Configures the range for depth bounds testing.

`func setFrontFacing(MTLWinding)`

Configures the vertex winding order that determines which face of a geometric primitive is the front one.

Required

`func setObjectThreadgroupMemoryLength(Int, index: Int)`

Configures the size of a threadgroup memory buffer for a threadgroup argument in the object shader function.

Required

`func setRenderPipelineState(any MTLRenderPipelineState)`

Configures this encoder with a render pipeline state that applies to your subsequent draw commands.

Required

`func setScissorRect(MTLScissorRect)`

Sets a scissor rectangle to discard fragments outside a specific area.

Required

`func setScissorRects([MTLScissorRect])`

Sets an array of scissor rectangles for a fragment scissor test.

`func setStencilReferenceValue(UInt32)`

Configures this encoder with a reference value for stencil testing.

Required

`func setStencilReferenceValue(front: UInt32, back: UInt32)`

Configures the encoder with different stencil test reference values for front-facing and back-facing primitives.

Required

`func setStencilStoreAction(MTLStoreAction)`

Configures the store action for the stencil attachment.

Required

`func setThreadgroupMemoryLength(Int, offset: Int, index: Int)`

Configures the size of a threadgroup memory buffer for a threadgroup argument in the fragment and tile shader functions.

Required

`func setTriangleFillMode(MTLTriangleFillMode)`

Configures how subsequent draw commands rasterize triangle and triangle strip primitives.

Required

`func setVertexAmplificationCount(Int)`

Sets the vertex amplification count and its view mapping for each amplification ID.

`func setVertexAmplificationCount([MTLVertexAmplificationViewMapping])`

Sets the vertex amplification count and its view mapping for each amplification ID.

`func setViewport(MTLViewport)`

Sets the viewport which that transforms vertices from normalized device coordinates to window coordinates.

Required

`func setViewports([MTLViewport])`

Sets an array of viewports to transform vertices from normalized device coordinates to window coordinates.

`func setVisibilityResultMode(MTLVisibilityResultMode, offset: Int)`

Configures a visibility test for Metal to run, and the destination for any results it generates.

Required

`func writeTimestamp(granularity: MTL4TimestampGranularity, after: MTLRenderStages, counterHeap: any MTL4CounterHeap, index: Int)`

Writes a GPU timestamp into the given `MTL4CounterHeap` at `index` after `stage` completes.

Required

Relationships

Inherits From

`MTL4CommandEncoder`, `NSObjectProtocol`

See Also

Encoding a render pass

`protocol MTLRenderCommandEncoder`

An interface that encodes a render pass into a command buffer, including all its draw calls and configuration.

`struct MTL4RenderEncoderOptions`

Custom render pass options you specify at encoder creation time.

`enum MTLTriangleFillMode`

Specifies how to rasterize triangle and triangle strip primitives.

`enum MTLWinding`

The vertex winding rule that determines a front-facing primitive.

`enum MTLCullMode`

The mode that determines whether to perform culling and which type of primitive to cull.

`enum MTLPrimitiveType`

The geometric primitive type for drawing commands.

`enum MTLIndexType`

The index type for an index buffer that references vertices of geometric primitives.

`enum MTLDepthClipMode`

The mode that determines how to deal with fragments outside of the near or far planes.

`enum MTLVisibilityResultMode`

The mode that determines what, if anything, the GPU writes to the results buffer, after the GPU executes the render pass.

`enum MTLVisibilityResultType`

This enumeration controls if Metal accumulates visibility results between render encoders or resets them.