API Collection

GPU devices and work submission

Find any available GPU, submit work to it with command buffers, suspend work, and coordinate between multiple GPUs.

Overview

You can use any available GPU's `MTLDevice` instance in addition to the default instance that `MTLCreateSystemDefaultDevice()` returns. For each device instance, get its `MTLCommandQueue` instance, and create one or more `MTLCommandBuffer` instances to send work to the GPU.

When the system suspends your app, use the command queue to finish command buffers already in progress. See Preparing your Metal app to run in the background for more information.

Topics

Locating and inspecting a GPU device

- Getting the default GPU
  Select the system's default GPU device on which to run your Metal code.
- Detecting GPU features and Metal software versions
  Use the device object's properties to determine how you perform tasks in Metal.

```
func MTLCreateSystemDefaultDevice() -> (any MTLDevice)?
```

Returns the device instance Metal selects as the default.

```
protocol MTLDevice
```

The main Metal interface to a GPU that apps use to draw graphics and run computations in parallel.

Multi-GPU systems

Locate and work with internal and external GPUs and their displays, video memory, and performance tradeoffs.

Submitting work to a GPU with Metal 4

`protocol MTL4CommandQueue`

An abstraction representing a command queue that you use commit and synchronize command buffers and to perform other GPU operations.

`class MTL4CommandQueueDescriptor`

Groups together parameters for the creation of a new command queue.

`struct MTL4CommandQueueError`

`enum Code`

Enumeration of kinds of errors that committing an array of command buffers instances can produce.

`let MTL4CommandQueueErrorDomain: String`

`protocol MTL4CommandBuffer`

Records a sequence of GPU commands.

`class MTL4CommandBufferOptions`

Options to configure a command buffer before encoding work into it.

`protocol MTL4CommandEncoder`

An encoder that writes GPU commands into a command buffer.

`struct MTL4RenderEncoderOptions`

Custom render pass options you specify at encoder creation time.

`protocol MTL4ArgumentTable`

Provides a mechanism to manage and provide resource bindings for buffers, textures, sampler states and other Metal resources.

`class MTL4ArgumentTableDescriptor`

Groups parameters for the creation of a Metal argument table.

`protocol` `MTL4CommandAllocator`

Manages the memory backing the encoding of GPU commands into command buffers.

`class` `MTL4CommandAllocatorDescriptor`

Groups together parameters for creating a command allocator.

`class` `MTL4CommitOptions`

Represents options to configure a commit operation on a command queue.

`protocol` `MTL4CommitFeedback`

Describes an object containing debug information from Metal to your app after completing a workload.

`typealias` `MTL4CommitFeedbackHandler`

Defines the block signature for a callback Metal invokes to provide your app feedback after completing a workload.

`protocol` `MTL4CounterHeap`

Represents an opaque, driver-controlled section of memory that can store GPU counter data.

`class` `MTL4CounterHeapDescriptor`

Groups together parameters for configuring a counter heap object at creation time.

`enum` `MTL4CounterHeapType`

Defines the type of a `MTL4CounterHeap` and the contents of its entries.

`struct` `MTL4TimestampHeapEntry`

Represents a timestamp data entry in a counter heap of type `MTL4CounterHeapType` `Timestamp`.

`enum` `MTL4TimestampGranularity`

Provides a hint to the system about the desired accuracy when writing GPU counter timestamps.

Submitting work to a GPU with Metal

Setting up a command structure

Discover how Metal executes commands on a GPU.

`protocol` `MTLCommandQueue`

An instance you use to create, submit, and schedule command buffers to a specific GPU device to run the commands within those buffers.

`class` `MTLCommandQueueDescriptor`

A configuration that customizes the behavior for a new command queue.

`protocol` `MTLCommandBuffer`

A container that stores a sequence of GPU commands that you encode into it.

`class` `MTLCommandBufferDescriptor`

A configuration that customizes the behavior for a new command buffer.

`struct` `MTLCommandBufferError`

The command buffer error codes that indicate why the GPU doesn't finish executing a command buffer.

`protocol` `MTLCommandEncoder`

An encoder that writes GPU commands into a command buffer.

Suspending work on a GPU

Preparing your Metal app to run in the background

Prepare your app to move into the background by pausing future GPU use and ensuring previous work is scheduled.