API Collection

Indirect command encoding

Store draw commands in Metal buffers and run them at a later time on the GPU, either once or repeatedly.

Overview

You can use an `MTLIndirectCommandBuffer` instance to store draw commands and invoke them at a later time. Metal executes all the draw commands in an indirect command buffer each time you submit it. This means you can use indirect command buffers multiple times, unlike `MTLCommandBuffer` instances, which are all single-use.

You can encode an indirect command buffer to run on either the CPU or the GPU. However, the GPU gives you the ability to immediately use the output of one pass as the input of a subsequent pass. For example, you can create an indirect command buffer with commands that conditionally draw visible items by running:

1. A compute kernel that identifies visible geometry and saves it to a result buffer
2. An indirect command buffer that uses the result buffer as its input to make decisions on what to draw

Topics

Indirect command buffers

Creating an indirect command buffer

Configure a descriptor to specify the properties of an indirect command buffer.

Specifying drawing and dispatch arguments indirectly

Use indirect commands if you don't know your draw or dispatch call arguments when you encode the command.

{} Encoding indirect command buffers on the CPU

Reduce CPU overhead and simplify your command execution by reusing commands.

{} Encoding indirect command buffers on the GPU

Maximize CPU to GPU parallelization by generating render commands on the GPU.

`protocol MTLIndirectCommandBuffer`

A command buffer containing reusable commands, encoded either on the CPU or GPU.

`class MTLIndirectCommandBufferDescriptor`

A configuration you create to customize an indirect command buffer.

`struct MTLIndirectCommandType`

The types of commands that you can encode into the indirect command buffer.

`struct MTLIndirectCommandBufferExecutionRange`

A range of commands in an indirect command buffer.

`func MTLIndirectCommandBufferExecutionRangeMake(UInt32, UInt32) -> MTLIndirectCommandBufferExecutionRange`

Creates a command execution range.

Indirect compute commands

`protocol MTLIndirectComputeCommand`

A compute command in an indirect command buffer.

`struct MTLRegion`

The bounds for a subset of an instance's elements.

`struct MTLSize`

A type that represents one, two, or three dimensions of a type instance, such as an array or texture.

`struct MTLOrigin`

The coordinates for the front upper-left corner of a region.

`struct MTLStageInRegionIndirectArguments`

The data layout required for the arguments needed to specify the stage-in region.

`struct MTLDispatchThreadgroupsIndirectArguments`

The data layout required for arguments needed to specify the size of threadgroups.

Render compute commands

`protocol MTLIndirectRenderCommand`

A render command in an indirect command buffer.

`struct MTLDrawPatchIndirectArguments`

The data layout required for drawing patches via indirect buffer calls.

`struct MTLDrawPrimitivesIndirectArguments`

The data layout required for drawing primitives via indirect buffer calls.

`struct MTLDrawIndexedPrimitivesIndirectArguments`

The data layout required for drawing indexed primitives via indirect buffer calls.

See Also

Command encoders

- Render passes

  Encode a render pass to draw graphics into an image.

- Compute passes

  Encode a compute pass that runs computations in parallel on a thread grid, processing and manipulating Metal resource data on multiple cores of a GPU.

- Machine-learning passes

  Add machine-learning model inference to your Metal app's GPU workflow.

- Blit passes

  Encode a block information transfer pass to adjust and copy data to and from GPU resources, such as buffers and textures.

- Ray tracing with acceleration structures

Build a representation of your scene's geometry using triangles and bounding volumes to quickly trace rays through the scene.