Structure

LayerRenderer.Drawable

A type that provides the textures and information you need to draw a frame of content.

macOS 26.0+ | visionOS 1.0+

```
struct Drawable
```

Mentioned in

Drawing fully immersive content using Metal

Overview

When you draw a frame of content, the frame's `LayerRenderer.Drawable` type provides the actual textures and rendering information you need. Do as much work as possible in advance to prepare for rendering, and retrieve the `LayerRenderer.Drawable` only when you're ready to start encoding commands into your Metal command buffers. The system recycles frames and their drawables for efficiency, so if you retrieve the drawable too early, it might not be ready to use.

Use the drawable's `LayerRenderer.Drawable.View` instances to determine where to draw your content in the provided textures. After you finish encoding your content, call `encodePresent(commandBuffer:)` to add a presentation notification to your command buffer. This command tells Compositor Services when to display the frame, and is essential for displaying your frame on time.

Topics

Getting the views

```
var views: [LayerRenderer.Drawable.View]
```

An array of viewports that tell you how to draw to the drawable's textures

```
struct View
```

A type that provides information on how to render content into the frame's textures.

Accessing the device orientation

```
var deviceAnchor: DeviceAnchor?
```

The device position and orientation you used to render the frame.

Getting the render textures

```
var colorTextures: [any MTLTexture]
```

An array of color textures to use to render the current frame.

```
var depthTextures: [any MTLTexture]
```

An array of depth textures to use to render the current frame.

Enqueueing a command buffer

```
func encodePresent(commandBuffer: any MTLCommandBuffer)
```

Encodes a notification event to the specified command buffer to present the drawable's content onscreen.

Getting the rasterization rate map

```
var rasterizationRateMaps: [any MTLRasterizationRateMap]
```

The rasterization rate maps to use when rendering the frame.

```
var flippedRasterizationRateMaps: [any MTLRasterizationRateMap]
```

The rasterization rate maps that are flipped around the y-axis.

Getting the projection matrix

```
enum AxisDirectionConvention
```

Constants that indicate the axis and direction to use for a perspective projection matrix.

Accessing pixel depth information

`var depthRange: simd_float2`

The distances to the far and near clipping planes from the person viewing the content, in meters.

Managing the state machine

`var state: LayerRenderer.Drawable.State`

The current operational state of a drawable instance.

`enum State`

The state of ownership for the drawable.

Synchronizing the drawing operation

`var frameTiming: LayerRenderer.Frame.Timing`

The timing information for the drawable’s frame.

`var presentationFrameIndex: CompositorFrameIndex`

The sequential index of a drawable’s frame.

Retrieving the target

`var target: LayerRenderer.Drawable.Target`

Returns a value that indicates the target of the drawable type.

`enum Target`

The target where the drawable will be displayed/used.

Creating a drawable

`init()`

Creates an uninitialized drawable.

Adding a render context

`struct RenderContext`

An object the compositer uses for rendering all effects associated with a layer renderer drawable.

`func addRenderContext(commandBuffer: any MTLCommandBuffer) -> LayerRenderer.Drawable.RenderContext`

Adds and returns a render context to a `LayerRenderer.Drawable` providing a metal command buffer.

`func addRenderContext() -> LayerRenderer.Drawable.RenderContext`

Adds and returns a render context to a `LayerRenderer.Drawable` that draws any content required by the compositor.

Structures

`struct TrackingArea`

Instance Properties

`var isContentCaptureProtected: Bool`

Returns whether content capture is protected and it is safe to draw content that should be protected from capture.

`var trackingAreasTextures: [any MTLTexture]`

Use the returned texture in your render pipeline to store the tracking areas ID used for hover effects and indirect gestures. The layer's texture topology determines the layout and content for each texture. The drawable's views contain information about how those views map to the textures.

Instance Methods

`func addTrackingArea(identifier: LayerRenderer.Drawable.TrackingArea.Identifier) -> LayerRenderer.Drawable.TrackingArea`

Returns a tracking area which is create on the drawable's list of tracking areas.

`func computeProjection(convention: AxisDirectionConvention, viewIndex: Int) -> matrix_float4x4`

`func encodePresent()`

Encodes a notification event to the specified command buffer to present the drawable’s content onscreen.

Relationships

Conforms To

`BitwiseCopyable`

See Also

Drawing environment

`struct View`

A type that provides information on how to render content into the frame’s textures.