Class

# MTLTextureDescriptor

An instance that you use to configure new Metal texture instances.

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

```
class MTLTextureDescriptor
```

## Mentioned in

- Choosing a resource storage mode for Apple GPUs
- Setting resource storage modes
- Synchronizing a managed resource in macOS
- Understanding the Metal 4 core API

## Overview

To create a new texture, first create an `MTLTextureDescriptor` instance and set its property values. Then, call either the `makeTexture(descriptor:)` or `makeTexture(descriptor:iosurface:plane:)` method of an `MTLDevice` instance, or the `makeTexture(descriptor:offset:bytesPerRow:)` method of an `MTLBuffer` instance.

When you create a texture, Metal copies property values from the descriptor into the new texture. You can reuse an `MTLTextureDescriptor` instance, modifying its property values as needed, to create more `MTLTexture` instances, without affecting any textures you already created.

## Topics

## Creating texture descriptors

```
class func texture2DDescriptor(pixelFormat: MTLPixelFormat, width: Int,
height: Int, mipmapped: Bool) -> MTLTextureDescriptor
```

Creates a texture descriptor object for a 2D texture.

```
class func textureCubeDescriptor(pixelFormat: MTLPixelFormat, size: Int
, mipmapped: Bool) -> MTLTextureDescriptor
```

Creates a texture descriptor object for a cube texture.

```
class func textureBufferDescriptor(with: MTLPixelFormat, width: Int,
resourceOptions: MTLResourceOptions, usage: MTLTextureUsage) ->
MTLTextureDescriptor
```

Creates a texture descriptor object for a texture buffer.

## Specifying texture attributes

```
var textureType: MTLTextureType
```

The dimension and arrangement of texture image data.

```
var pixelFormat: MTLPixelFormat
```

The size and bit layout of all pixels in the texture.

```
var width: Int
```

The width of the texture image for the base level mipmap, in pixels.

```
var height: Int
```

The height of the texture image for the base level mipmap, in pixels.

```
var depth: Int
```

The depth of the texture image for the base level mipmap, in pixels.

```
var mipmapLevelCount: Int
```

The number of mipmap levels for this texture.

```
var sampleCount: Int
```

The number of samples in each fragment.

```
var arrayLength: Int
```

The number of array elements for this texture.

`var resourceOptions: MTLResourceOptions`

The behavior of a new memory allocation.

`var cpuCacheMode: MTLCPUCacheMode`

The CPU cache mode used for the CPU mapping of the texture.

`var storageMode: MTLStorageMode`

The location and access permissions of the texture.

`var hazardTrackingMode: MTLHazardTrackingMode`

The texture’s hazard tracking mode.

`var allowGPUOptimizedContents: Bool`

A Boolean value indicating whether the GPU is allowed to adjust the texture’s contents to improve GPU performance.

`var usage: MTLTextureUsage`

Options that determine how you can use the texture.

`var swizzle: MTLTextureSwizzleChannels`

The pattern you want the GPU to apply to pixels when you read or sample pixels from the texture.

`struct MTLTextureSwizzleChannels`

A pattern that modifies the data read or sampled from a texture by rearranging or duplicating the elements of a vector.

`enum MTLTextureSwizzle`

A set of options to choose from when creating a texture swizzle pattern.

`enum MTLTextureType`

The dimension of each image, including whether multiple images are arranged into an array or a cube.

`struct MTLTextureUsage`

An enumeration for the various options that determine how you can use a texture.

## Instance Properties

`var compressionType: MTLTextureCompressionType`

`var placementSparsePageSize: MTLSparsePageSize`

Determines the page size for a placement sparse texture.

## Relationships

### Inherits From

`NSObject`

### Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSCopying`
`NSObjectProtocol`

## See Also

### Texture basics

Understanding color-renderable pixel format sizes

Know the size limits of color render targets in Apple GPUs based on the target’s pixel format.

Optimizing texture data

Optimize a texture’s data to improve GPU or CPU access.

`protocol MTLTexture`

A resource that holds formatted image data.

`enum MTLTextureCompressionType`

`class MTKTextureLoader`

An object that creates textures from existing data in common image formats.

`class MTLSharedTextureHandle`

A texture handle that can be shared across process address space boundaries.

`enum MTLPixelFormat`

The data formats that describe the organization and characteristics of individual pixels in a texture.