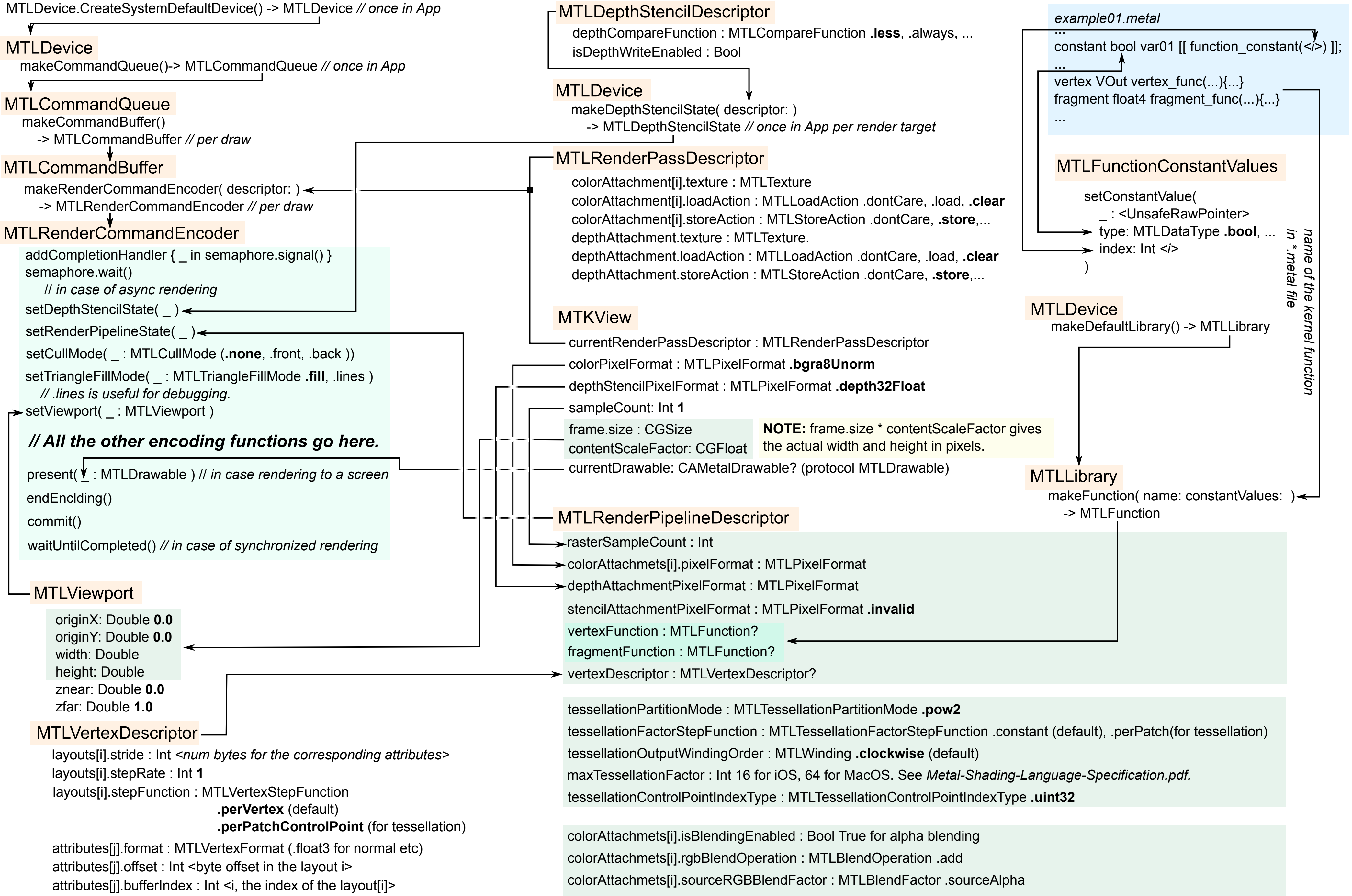
Initialization & Configuration



MTLDevice.CreateSystemDefaultDevice() -> MTLDevice *// once in App*

MTLDevice
makeCommandQueue()-> MTLCommandQueue *// once in App*

MTLCommandQueue
makeCommandBuffer()
-> MTLCommandBuffer *// per draw*

MTLCommandBuffer
makeRenderCommandEncoder(descriptor:)
-> MTLRenderCommandEncoder *// per draw*

MTLRenderCommandEncoder
addCompletionHandler { _ in semaphore.signal() }
semaphore.wait()
// in case of async rendering
setDepthStencilState(_)
setRenderPipelineState(_)
setCullMode(_ : MTLCullMode (**.none**, .front, .back))
setTriangleFillMode(_ : MTLTriangleFillMode **.fill**, .lines)
// .lines is useful for debugging.
setViewport(_ : MTLViewport)

// All the other encoding functions go here.

present(_ : MTLDrawable) // *in case rendering to a screen*
endEnclding()
commit()
waitUntilCompleted() *// in case of synchronized rendering*

MTLViewport
originX: Double **0.0**
originY: Double **0.0**
width: Double
height: Double
znear: Double **0.0**
zfar: Double **1.0**

MTLVertexDescriptor
layouts[i].stride : Int *<num bytes for the corresponding attributes>*
layouts[i].stepRate : Int **1**
layouts[i].stepFunction : MTLVertexStepFunction
.perVertex (default)
.perPatchControlPoint (for tessellation)
attributes[j].format : MTLVertexFormat (.float3 for normal etc)
attributes[j].offset : Int <byte offset in the layout i>
attributes[j].bufferIndex : Int <i, the index of the layout[i]>

MTLDepthStencilDescriptor
depthCompareFunction : MTLCompareFunction **.less**, .always, ...
isDepthWriteEnabled : Bool

MTLDevice
makeDepthStencilState(descriptor:)
-> MTLDepthStencilState *// once in App per render target*

MTLRenderPassDescriptor
colorAttachment[i].texture : MTLTexture
colorAttachment[i].loadAction : MTLLoadAction .dontCare, .load, **.clear**
colorAttachment[i].storeAction : MTLStoreAction .dontCare, **.store**,...
depthAttachment.texture : MTLTexture.
depthAttachment.loadAction : MTLLoadAction .dontCare, .load, **.clear**
depthAttachment.storeAction : MTLStoreAction .dontCare, **.store**,...

MTKView
currentRenderPassDescriptor : MTLRenderPassDescriptor
colorPixelFormat : MTLPixelFormat **.bgra8Unorm**
depthStencilPixelFormat : MTLPixelFormat **.depth32Float**
sampleCount: Int **1**
frame.size : CGSize
contentScaleFactor: CGFloat
currentDrawable: CAMetalDrawable? (protocol MTLDrawable)

NOTE: frame.size * contentScaleFactor gives the actual width and height in pixels.

MTLRenderPipelineDescriptor
rasterSampleCount : Int
colorAttachmets[i].pixelFormat : MTLPixelFormat
depthAttachmentPixelFormat : MTLPixelFormat
stencilAttachmentPixelFormat : MTLPixelFormat **.invalid**
vertexFunction : MTLFunction?
fragmentFunction : MTLFunction?
vertexDescriptor : MTLVertexDescriptor?

tessellationPartitionMode : MTLTessellationPartitionMode **.pow2**
tessellationFactorStepFunction : MTLTessellationFactorStepFunction .constant (default), .perPatch(for tessellation)
tessellationOutputWindingOrder : MTLWinding **.clockwise** (default)
maxTessellationFactor : Int 16 for iOS, 64 for MacOS. See *Metal-Shading-Language-Specification.pdf.*
tessellationControlPointIndexType : MTLTessellationControlPointIndexType **.uint32**

colorAttachmets[i].isBlendingEnabled : Bool True for alpha blending
colorAttachmets[i].rgbBlendOperation : MTLBlendOperation .add
colorAttachmets[i].sourceRGBBlendFactor : MTLBlendFactor .sourceAlpha
colorAttachmets[i].destinationRGBBlendFactor : MTLBlendFactor .oneMinusSourceAlpha

example01.metal

```
...
constant bool var01 [[ function_constant(<i>) ]];
...
vertex VOut vertex_func(...){...}
fragment float4 fragment_func(...){...}
...
```

MTLFunctionConstantValues
setConstantValue(
_ : <UnsafeRawPointer>
type: MTLDataType **.bool**, ...
index: Int *<i>*
)

MTLDevice
makeDefaultLibrary() -> MTLLibrary

*name of the kernel function in *.metal file*

MTLLibrary
makeFunction(name: constantValues:)
-> MTLFunction

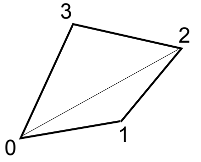
Drawing Triangles & Patches, and the Indices

Drawing Triangles

```
drawIndexedPrimitives(
    type: MTLPrimitiveType .triangle, .line, point,...
    indexCount: Int <number of indices (number of triangles x 3)>
    indexType: MTLIndexType .uint32, uint16
    indexBuffer: MTLBuffer,
    indexBufferOffset: Int 0
    instanceCount: Int <number of instances>
    baseVertex: Int 0
    baseInstance: Int 0
)
```

MTLBuffer [UInt32]

0
1
2
0
2
3



Drawing Patches

```
setTessellationFactorBuffer(
    - : MTLBuffer?
    offset: 0
    instanceStride: 0
)
```

MTLBuffer [HalfFloat]

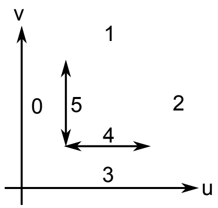
length: 9 x 6

0	4.0
1	2.0
2	2.0
3	4.0
4	4.0
5	4.0
	⋮

metal_tessellation.h

```
...
struct MTLQuadTessellationFactorsHalf
{
    half edgeTessellationFactor[4];
    half insideTessellationFactor[2];
};
...
```

Mapping from the indices to the edge & inside factors



```
drawIndexedPatches(
    numberOfPatchControlPoints: Int <number of control points per patch, ex. 16>
    patchIndexBuffer: MTLBuffer? nil
    patchIndexBufferOffset: Int 0
    controlPointIndexBuffer: MTLBuffer,
    controlPointIndexBufferOffset: Int 0
    indirectBuffer: MTLBuffer,
    indirectBufferOffset: Int 0
)
```

MTLDrawPatchIndirectArguments

```
patchCount: UInt32 <number of patches, ex. 9>
instanceCount: UInt32 1
patchStart: UInt32 0
baseInstance: UInt32 0
```

MTLBuffer [UInt32]

length: 9 x 16

0
1
2
3
6
7
8
9
12
13
14
15
18
19
20
21
1
2
3
4
7
8
9
10
⋮
35

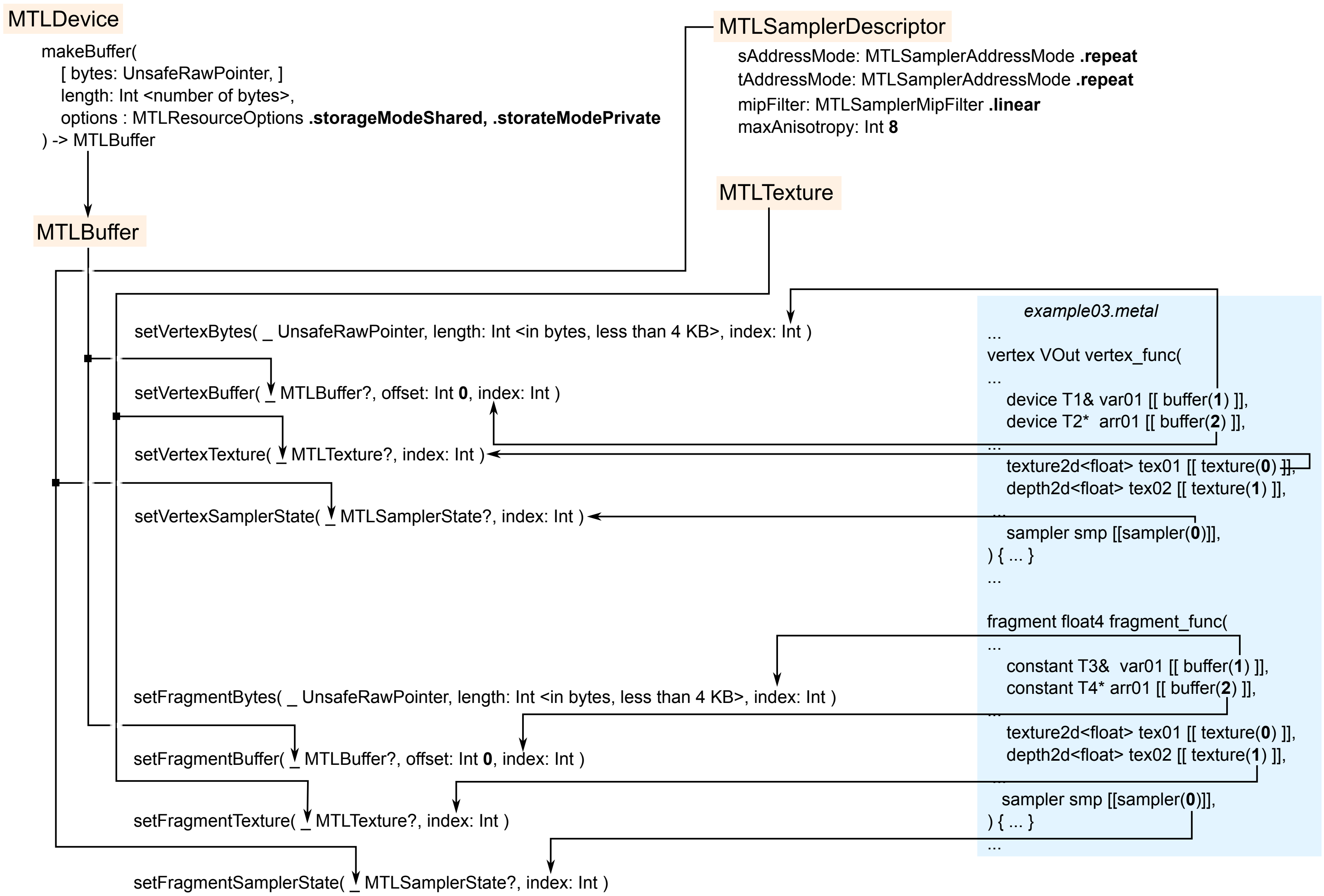
Control Points in a Grid
9 active patches

30	31	32	33	34	35
24	25	**26**	**27**	**28**	29
18	19	**20**	**21**	**22**	23
12	13	**14**	**15**	**16**	17
6	7	8	9	10	11
0	1	2	3	4	5

example02.metal

```
...
typedef struct _TCP {
    float4 p [[ attribute(0) ]];
} TCP;
...
[[ patch( quad, 16 ) ]] vertex VOut vertex_func(
    patch_control_point<TCP> cps [[ stage_in ]],
    ...
){...}
...
```

Assignment of the Parameters to the Vertex & Fragment Shaders



MTLDevice

```
makeBuffer(
   [ bytes: UnsafeRawPointer, ]
   length: Int <number of bytes>,
   options : MTLResourceOptions .storageModeShared, .storateModePrivate
) -> MTLBuffer
```

MTLBuffer

MTLSamplerDescriptor

sAddressMode: MTLSamplerAddressMode **.repeat**
tAddressMode: MTLSamplerAddressMode **.repeat**
mipFilter: MTLSamplerMipFilter **.linear**
maxAnisotropy: Int **8**

MTLTexture

setVertexBytes(_ UnsafeRawPointer, length: Int <in bytes, less than 4 KB>, index: Int)

setVertexBuffer(_ MTLBuffer?, offset: Int **0**, index: Int)

setVertexTexture(_ MTLTexture?, index: Int)

setVertexSamplerState(_ MTLSamplerState?, index: Int)

setFragmentBytes(_ UnsafeRawPointer, length: Int <in bytes, less than 4 KB>, index: Int)

setFragmentBuffer(_ MTLBuffer?, offset: Int **0**, index: Int)

setFragmentTexture(_ MTLTexture?, index: Int)

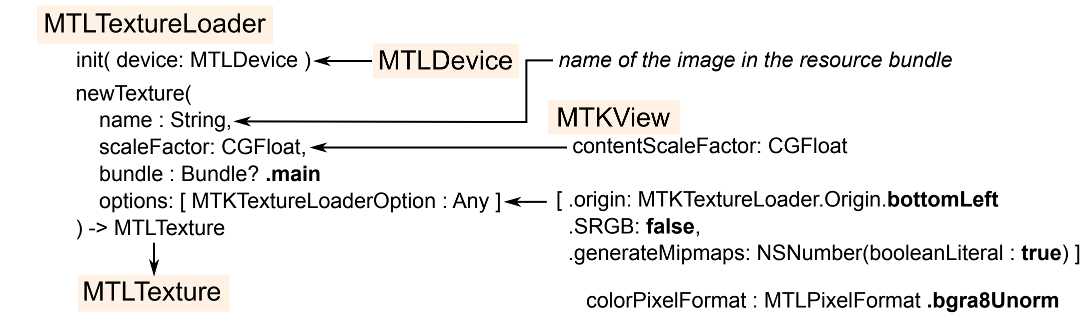
setFragmentSamplerState(_ MTLSamplerState?, index: Int)

example03.metal

```
...
vertex VOut vertex_func(
...
   device T1& var01 [[ buffer(1) ]],
   device T2*  arr01 [[ buffer(2) ]],
...
   texture2d<float> tex01 [[ texture(0) ]],
   depth2d<float> tex02 [[ texture(1) ]],
 ...
   sampler smp [[sampler(0)]],
) { ... }
...

fragment float4 fragment_func(
...
   constant T3&  var01 [[ buffer(1) ]],
   constant T4* arr01 [[ buffer(2) ]],
...
   texture2d<float> tex01 [[ texture(0) ]],
   depth2d<float> tex02 [[ texture(1) ]],
 ...
  sampler smp [[sampler(0)]],
) { ... }
...
```

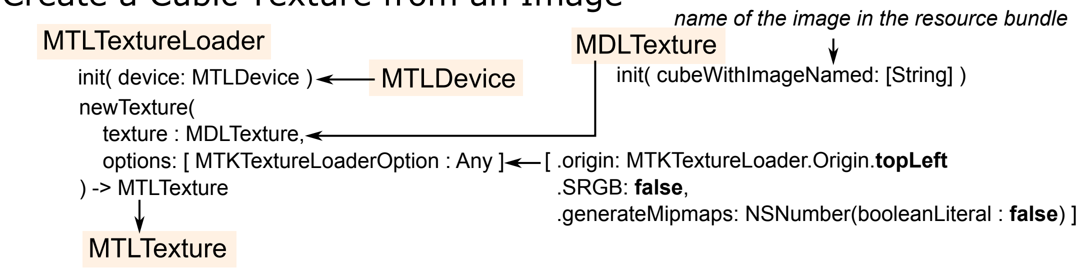
Texture Generation

Create a Texture from an Image



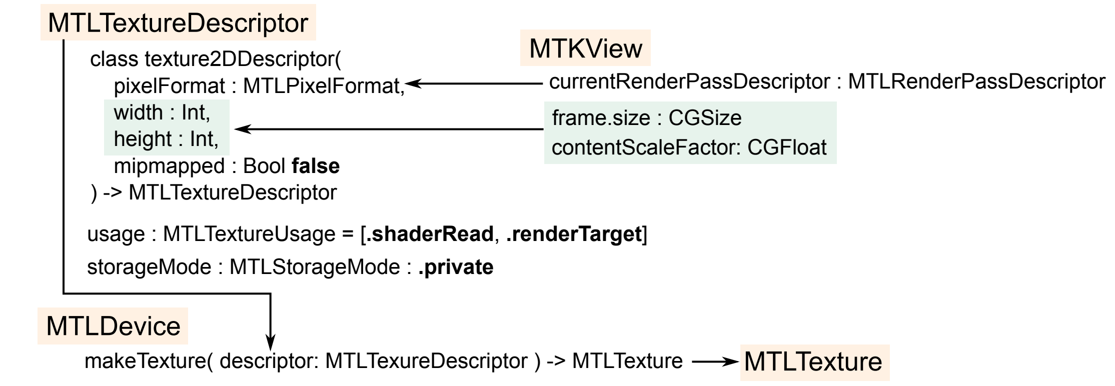
```
MTLTextureLoader
  init( device: MTLDevice ) <- MTLDevice
  newTexture(
    name : String,                       <- name of the image in the resource bundle
    scaleFactor: CGFloat,                <- MTKView contentScaleFactor: CGFloat
    bundle : Bundle? .main
    options: [ MTKTextureLoaderOption : Any ] <- [ .origin: MTKTextureLoader.Origin.bottomLeft
                                                  .SRGB: false,
                                                  .generateMipmaps: NSNumber(booleanLiteral : true) ]
  ) -> MTLTexture
MTLTexture

colorPixelFormat : MTLPixelFormat .bgra8Unorm
```

Create a Cubic Texture from an Image



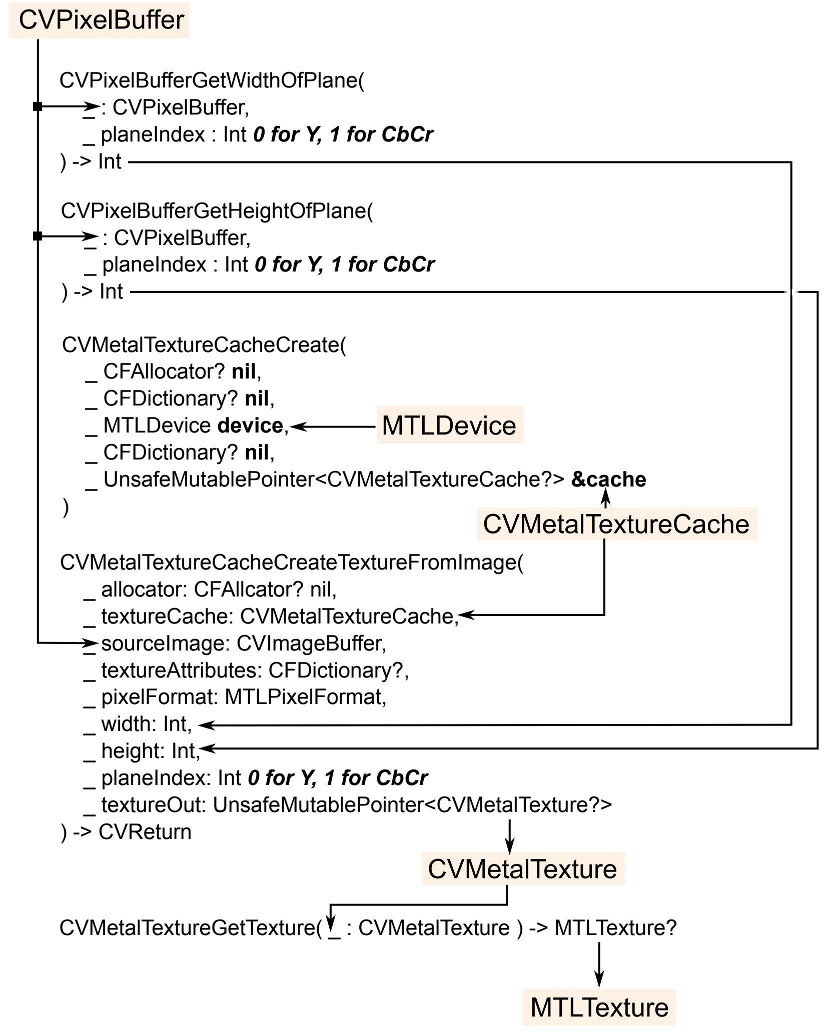
```
MTLTextureLoader
  init( device: MTLDevice ) <- MTLDevice
  newTexture(
    texture : MDLTexture,                <- MDLTexture
    options: [ MTKTextureLoaderOption : Any ] <- [ .origin: MTKTextureLoader.Origin.topLeft
                                                  .SRGB: false,
                                                  .generateMipmaps: NSNumber(booleanLiteral : false) ]
  ) -> MTLTexture
MTLTexture

MDLTexture
  init( cubeWithImageNamed: [String] )  <- name of the image in the resource bundle
```

Create an Empty Texture



```
MTLTextureDescriptor
  class texture2DDescriptor(
    pixelFormat : MTLPixelFormat,   <- MTKView currentRenderPassDescriptor : MTLRenderPassDescriptor
    width : Int,
    height : Int,                   <- frame.size : CGSize
                                       contentScaleFactor: CGFloat
    mipmapped : Bool false
  ) -> MTLTextureDescriptor
  usage : MTLTextureUsage = [.shaderRead, .renderTarget]
  storageMode : MTLStorageMode : .private

MTLDevice
  makeTexture( descriptor: MTLTexureDescriptor ) -> MTLTexture -> MTLTexture
```

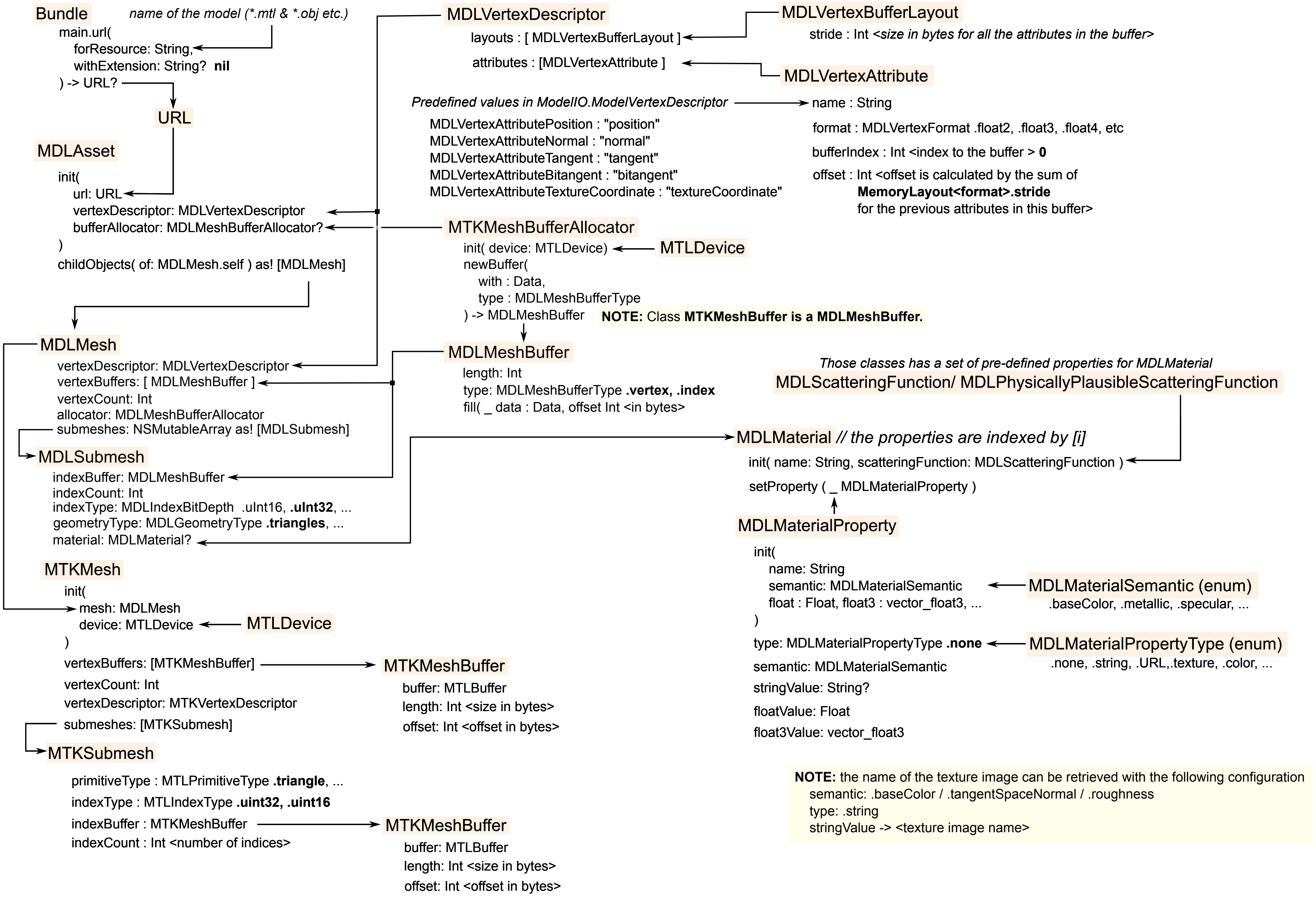
Create a Texture from Core Video Images



```
CVPixelBuffer

CVPixelBufferGetWidthOfPlane(
  _: CVPixelBuffer,
  _ planeIndex : Int 0 for Y, 1 for CbCr
) -> Int

CVPixelBufferGetHeightOfPlane(
  _: CVPixelBuffer,
  _ planeIndex : Int 0 for Y, 1 for CbCr
) -> Int

CVMetalTextureCacheCreate(
  _ CFAllocator? nil,
  _ CFDictionary? nil,
  _ MTLDevice device,  <- MTLDevice
  _ CFDictionary? nil,
  _ UnsafeMutablePointer<CVMetalTextureCache?> &cache
)
CVMetalTextureCache

CVMetalTextureCacheCreateTextureFromImage(
  _ allocator: CFAllcator? nil,
  _ textureCache: CVMetalTextureCache,
  _ sourceImage: CVImageBuffer,
  _ textureAttributes: CFDictionary?,
  _ pixelFormat: MTLPixelFormat,
  _ width: Int,
  _ height: Int,
  _ planeIndex: Int 0 for Y, 1 for CbCr
  _ textureOut: UnsafeMutablePointer<CVMetalTexture?>
) -> CVReturn
CVMetalTexture

CVMetalTextureGetTexture( _ : CVMetalTexture ) -> MTLTexture?
MTLTexture
```

NOTE: CVImageBuffer == CVPixelBuffer == CVBuffer

MDLMesh and MTKMesh



Bundle *name of the model (*.mtl & *.obj etc.)*

```
main.url(
   forResource: String,
   withExtension: String?  nil
) -> URL?
```

URL

MDLAsset

```
init(
   url: URL
   vertexDescriptor: MDLVertexDescriptor
   bufferAllocator: MDLMeshBufferAllocator?
)
childObjects( of: MDLMesh.self ) as! [MDLMesh]
```

MDLMesh

```
vertexDescriptor: MDLVertexDescriptor
vertexBuffers: [ MDLMeshBuffer ]
vertexCount: Int
allocator: MDLMeshBufferAllocator
submeshes: NSMutableArray as! [MDLSubmesh]
```

MDLSubmesh

```
indexBuffer: MDLMeshBuffer
indexCount: Int
indexType: MDLIndexBitDepth  .uInt16, .uInt32, ...
geometryType: MDLGeometryType .triangles, ...
material: MDLMaterial?
```

MTKMesh

```
init(
   mesh: MDLMesh
   device: MTLDevice
)
vertexBuffers: [MTKMeshBuffer]
vertexCount: Int
vertexDescriptor: MTKVertexDescriptor
submeshes: [MTKSubmesh]
```

MTLDevice

MTKMeshBuffer

```
buffer: MTLBuffer
length: Int <size in bytes>
offset: Int <offset in bytes>
```

MTKSubmesh

```
primitiveType : MTLPrimitiveType .triangle, ...
indexType : MTLIndexType .uint32, .uint16
indexBuffer : MTKMeshBuffer
indexCount : Int <number of indices>
```

MTKMeshBuffer

```
buffer: MTLBuffer
length: Int <size in bytes>
offset: Int <offset in bytes>
```

MDLVertexDescriptor

```
layouts : [ MDLVertexBufferLayout ]
attributes : [MDLVertexAttribute ]
```

MDLVertexBufferLayout

stride : Int <*size in bytes for all the attributes in the buffer*>

MDLVertexAttribute

Predefined values in ModelIO.ModelVertexDescriptor

```
MDLVertexAttributePosition : "position"
MDLVertexAttributeNormal : "normal"
MDLVertexAttributeTangent : "tangent"
MDLVertexAttributeBitangent : "bitangent"
MDLVertexAttributeTextureCoordinate : "textureCoordinate"
```

```
name : String
format : MDLVertexFormat .float2, .float3, .float4, etc
bufferIndex : Int <index to the buffer > 0
offset : Int <offset is calculated by the sum of
        MemoryLayout<format>.stride
        for the previous attributes in this buffer>
```

MTKMeshBufferAllocator

```
init( device: MTLDevice)
newBuffer(
   with : Data,
   type : MDLMeshBufferType
) -> MDLMeshBuffer
```

MTLDevice

NOTE: Class **MTKMeshBuffer is a MDLMeshBuffer.**

MDLMeshBuffer

```
length: Int
type: MDLMeshBufferType .vertex, .index
fill( _ data : Data, offset Int <in bytes>
```

Those classes has a set of pre-defined properties for MDLMaterial

MDLScatteringFunction/ MDLPhysicallyPlausibleScatteringFunction

MDLMaterial *// the properties are indexed by [i]*

```
init( name: String, scatteringFunction: MDLScatteringFunction )
setProperty ( _ MDLMaterialProperty )
```

MDLMaterialProperty

```
init(
   name: String
   semantic: MDLMaterialSemantic
   float : Float, float3 : vector_float3, ...
)
type: MDLMaterialPropertyType .none
semantic: MDLMaterialSemantic
stringValue: String?
floatValue: Float
float3Value: vector_float3
```

MDLMaterialSemantic (enum)

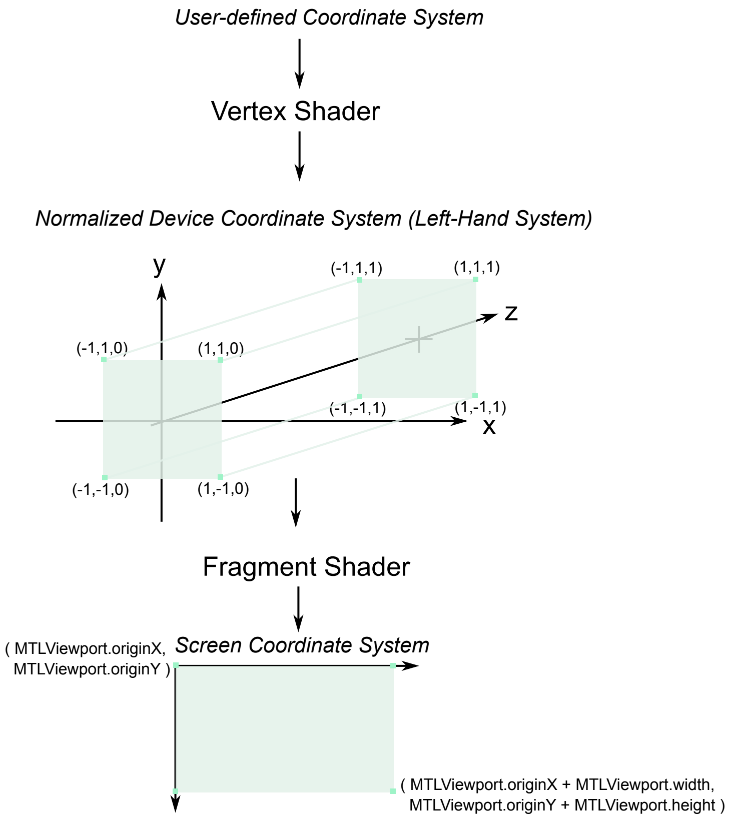
.baseColor, .metallic, .specular, ...

MDLMaterialPropertyType (enum)

.none, .string, .URL,.texture, .color, ...

NOTE: the name of the texture image can be retrieved with the following configuration
semantic: .baseColor / .tangentSpaceNormal / .roughness
type: .string
stringValue -> <texture image name>

Coordinate Systems and Others



User-defined Coordinate System

Vertex Shader

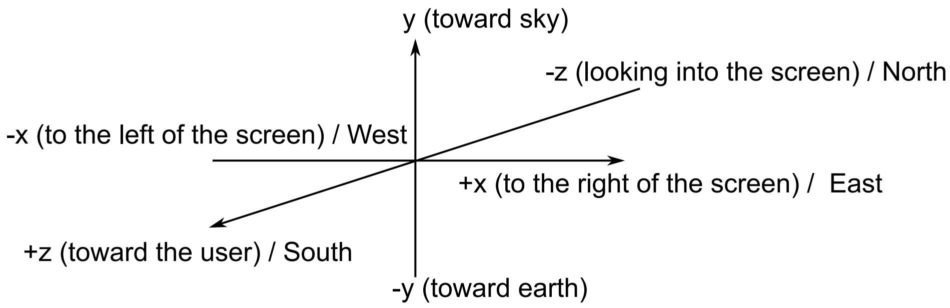
Normalized Device Coordinate System (Left-Hand System)

Fragment Shader

Screen Coordinate System

y
z
x
(-1,1,1)
(1,1,1)
(-1,1,0)
(1,1,0)
(-1,-1,1)
(1,-1,1)
(-1,-1,0)
(1,-1,0)
(MTLViewport.originX, MTLViewport.originY)
(MTLViewport.originX + MTLViewport.width, MTLViewport.originY + MTLViewport.height)

ARKit Coordinate System



y (toward sky)
-z (looking into the screen) / North
-x (to the left of the screen) / West
+x (to the right of the screen) / East
+z (toward the user) / South
-y (toward earth)

How to discard a vertex in the vertex shader

```
...
struct VOClip {
   float4 p [[ position ]];
...
   float c [[ clip_distance ]] [1];
};
// The only difference from COVlip
// is the absence of float c.
struct VO {
   float4 p [[ position ]];
...
}
vertex VOClip vertex_func(...){
...
   VOClip out {
      .p = position,
      ...
      .c = clip_distance // if negative, the vertex is discarded.
   return out;
}
...
fragment float4 fragment_func(
   VO in [[ stage_in ]], // This is not VOClip, but VO.
...) { ... }
```

How to discard a vertex in the fragment shader

```
...
// Just call discard_fragment(); in the fragment shader.
fragment float4 fragment_func(...) {
   ...
   if (discard) {
      discard_fragment();
   }
}
```