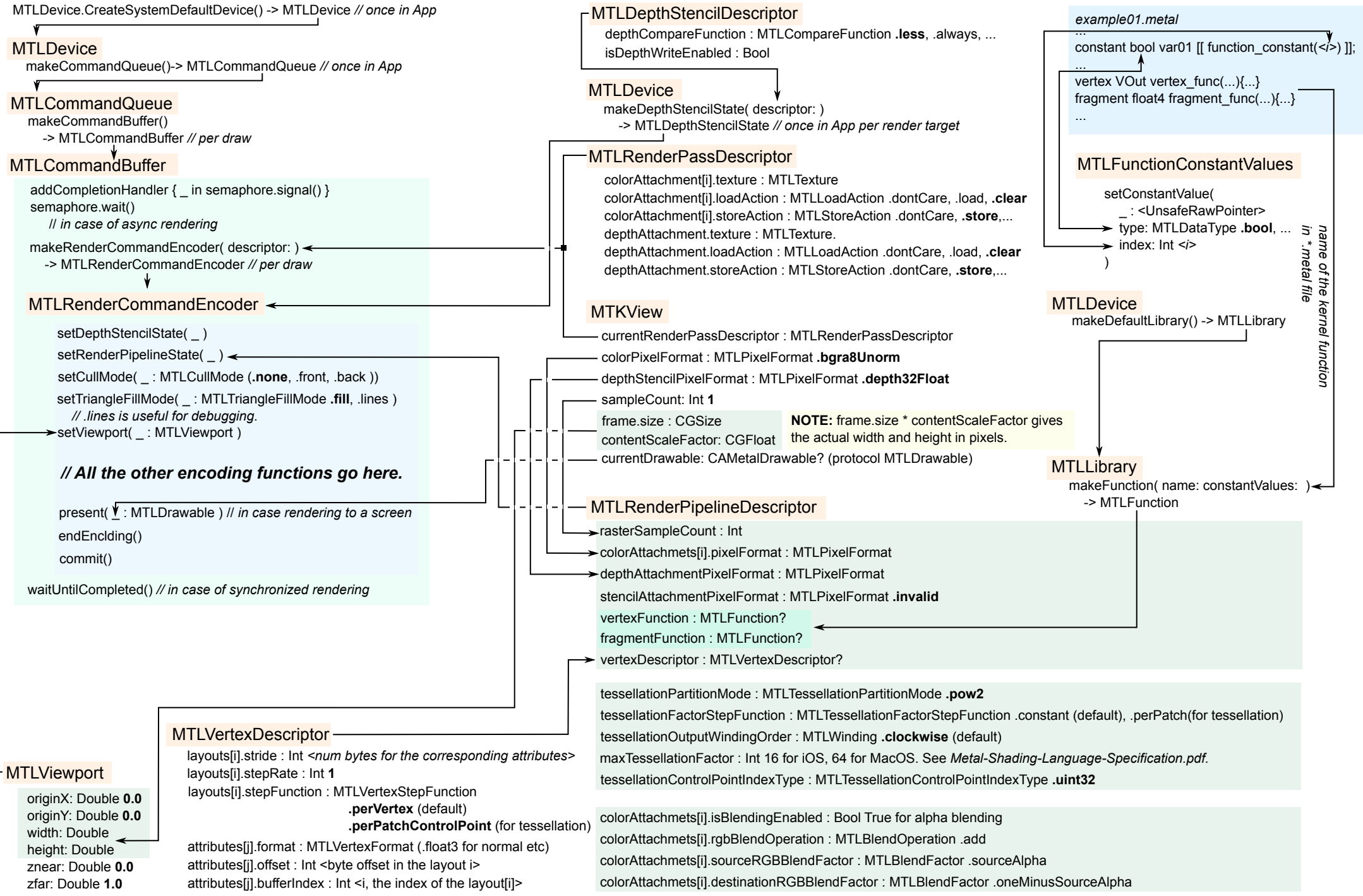
Initialization & Configuration



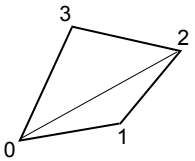
Drawing Triangles & Patches, and the Indices

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]

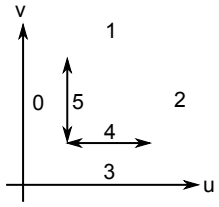
length: 9 x 6

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
)
```

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

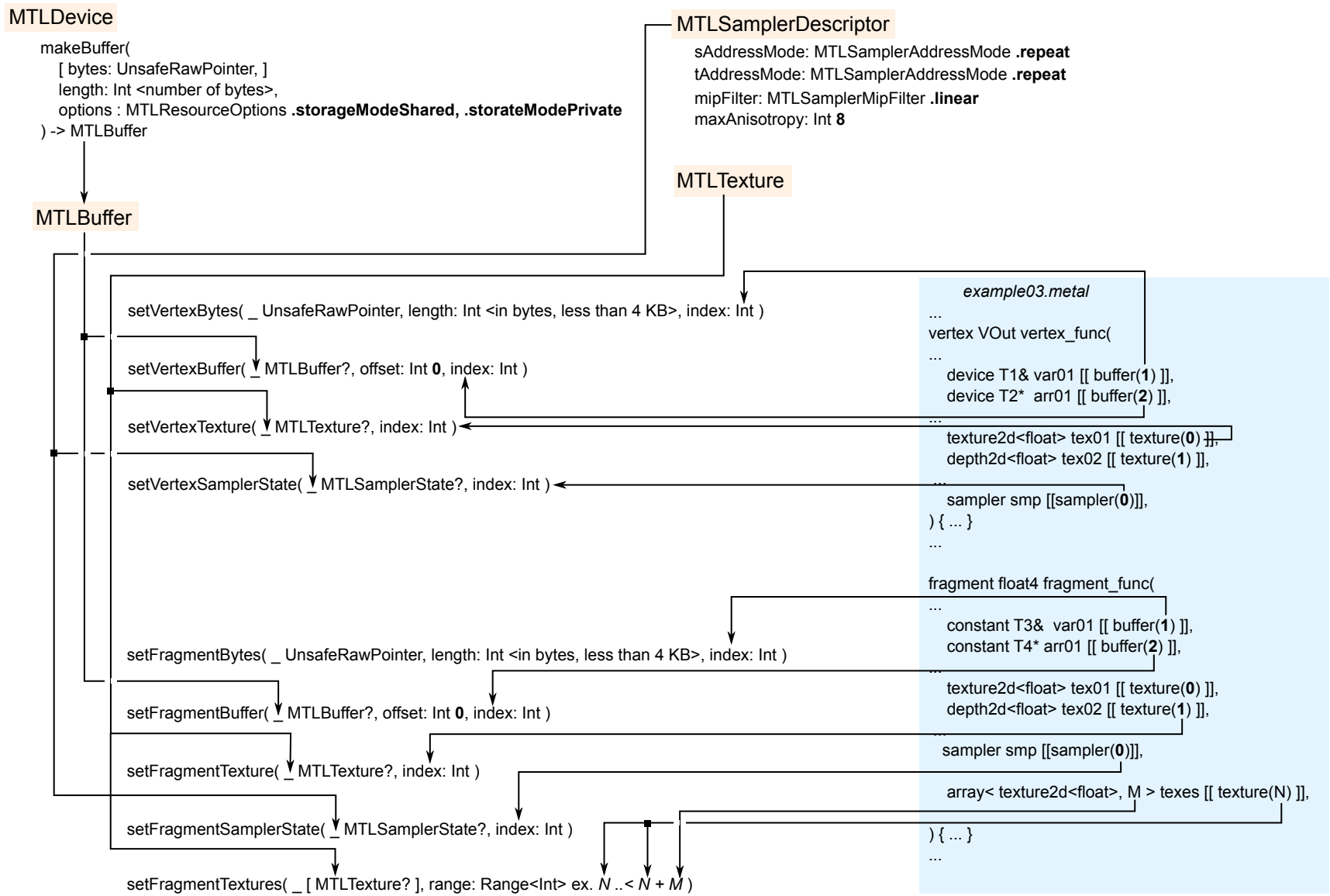
MTLDrawPatchIndirectArguments

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

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

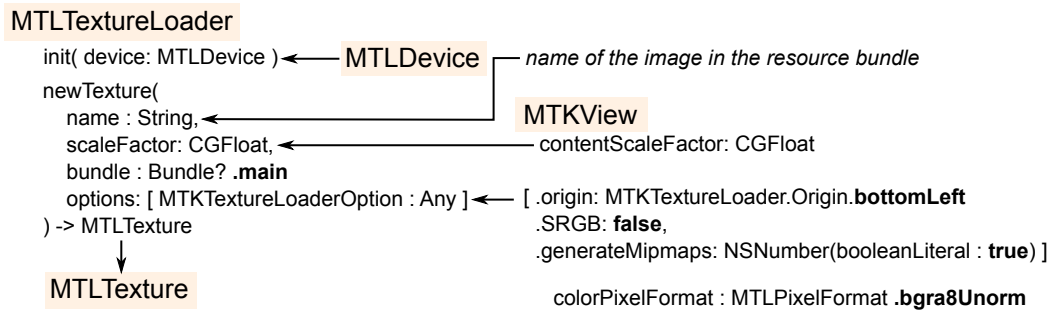


NOTE: Multiple textures of variable numbers M with the starting index N can be specified with the following pair of APIs.
- setFragmentTextures() in Swift.
- array< texture2d<float>, M> texes [[...]] in the formal parameters of the fragment shader.

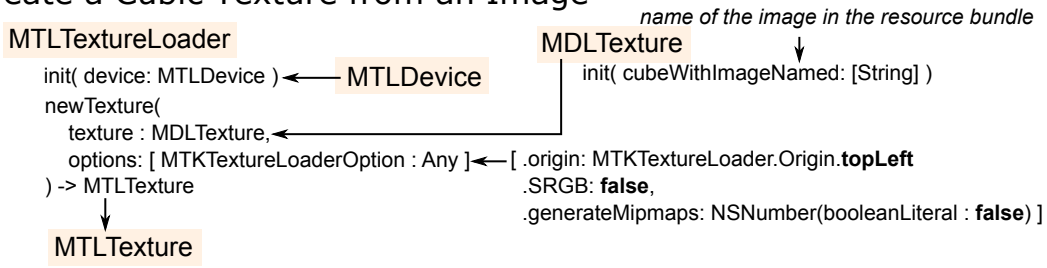
M must be determined at the compilation time. However, **at runtime, not all the M elements must be specified in [MTLTexture?].**
In the fragment shader each texture can be accessed by array indexing, i.e., texes[N+i].

Texture Generation

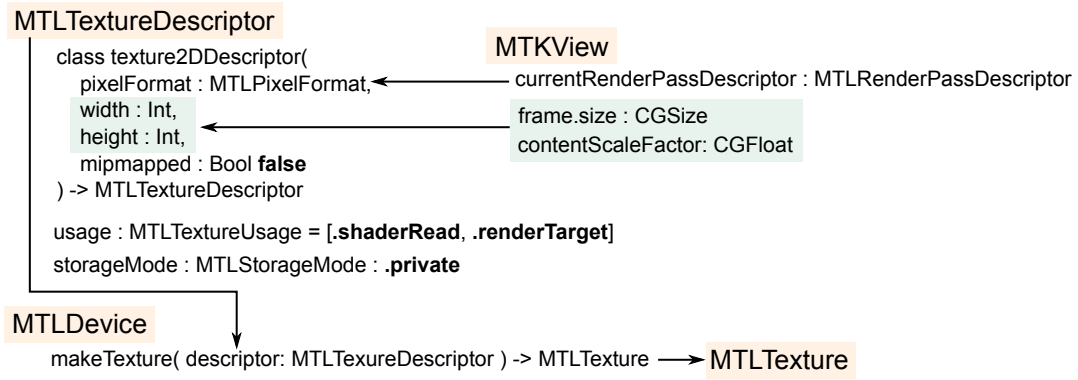
Create a Texture from an Image



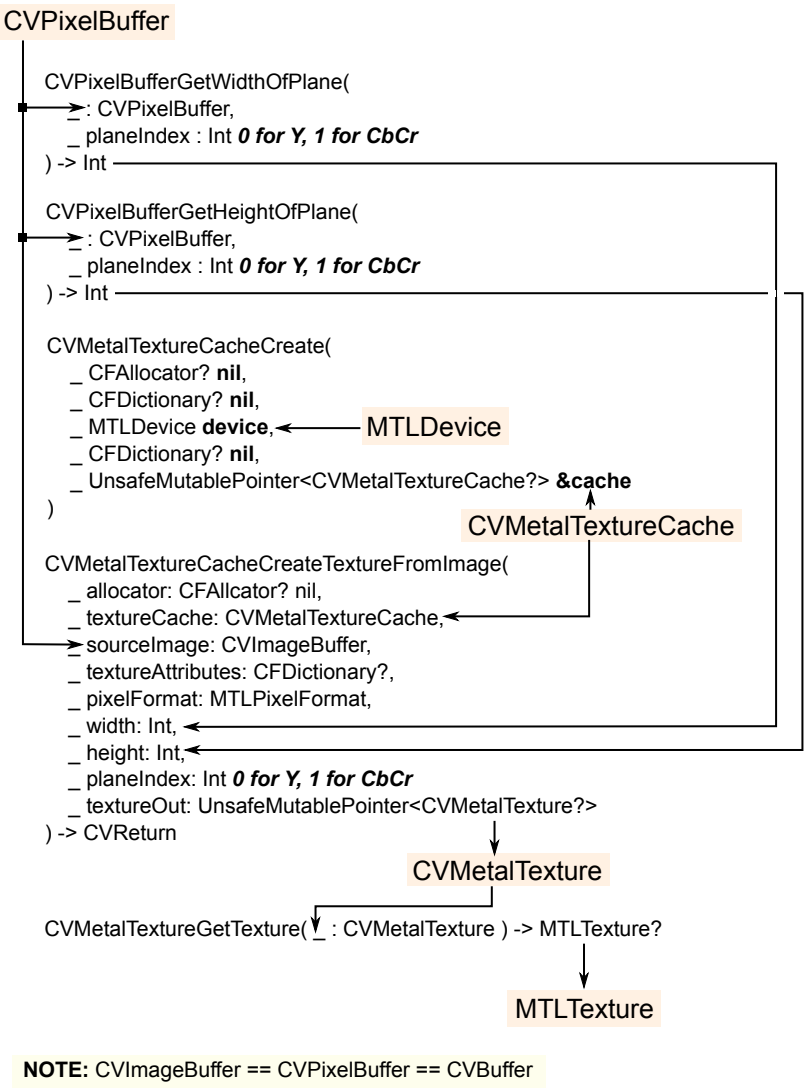
Create a Cubic Texture from an Image



Create an Empty Texture

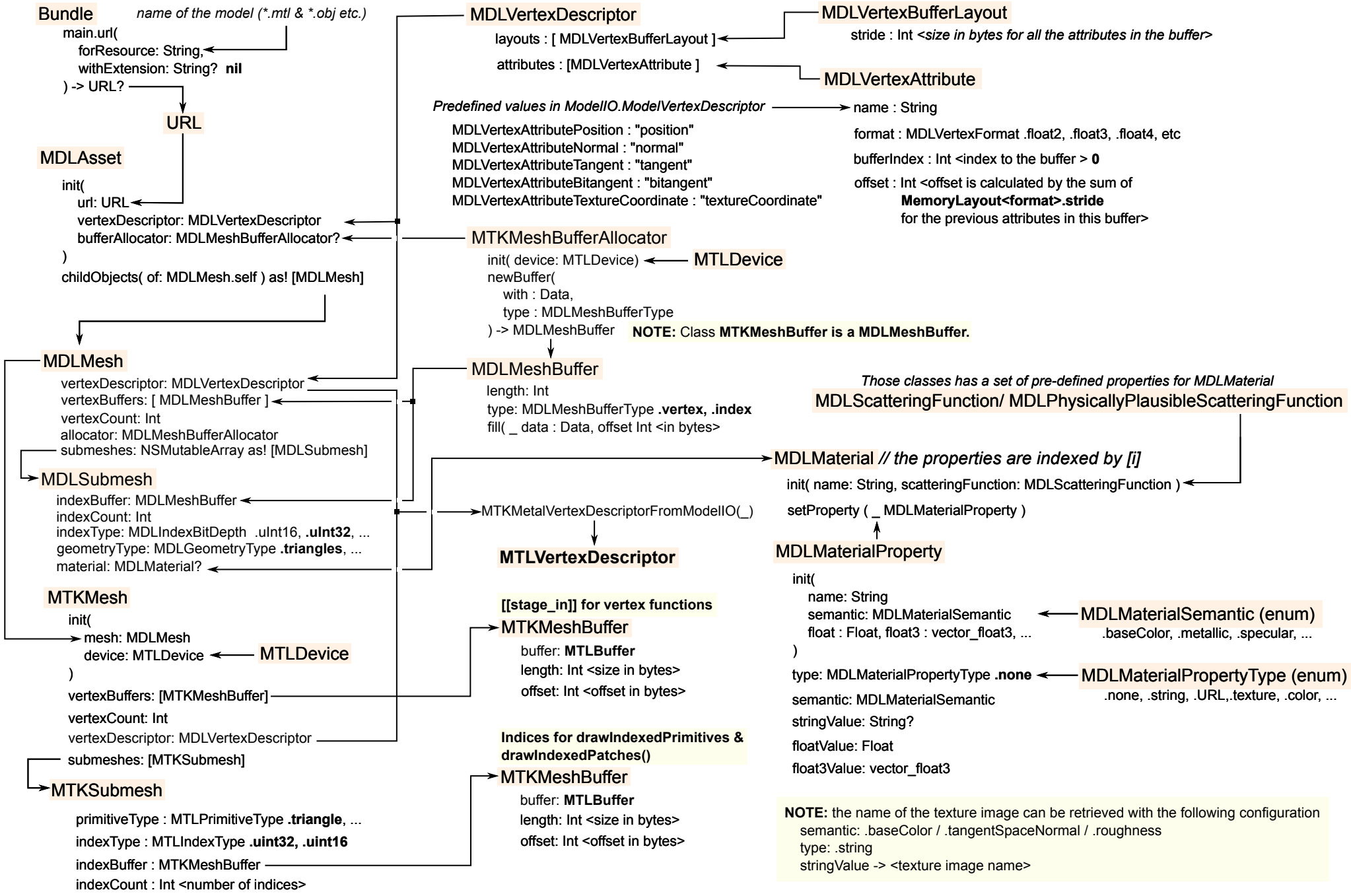


Create a Texture from Core Video Images

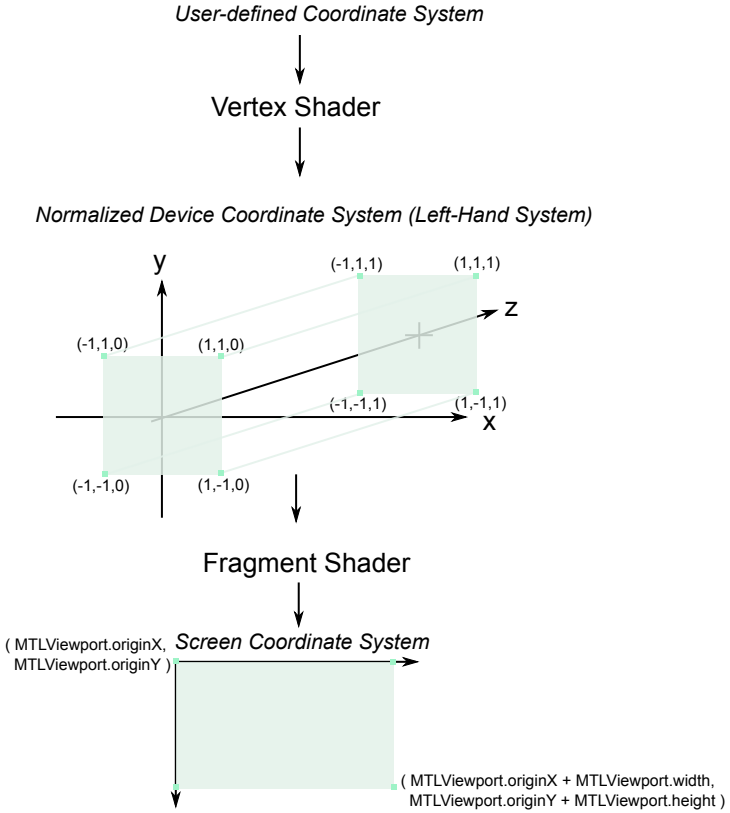


NOTE: CVImageBuffer == CVPixelBuffer == CVBuffer

MDLMesh and MTKMesh



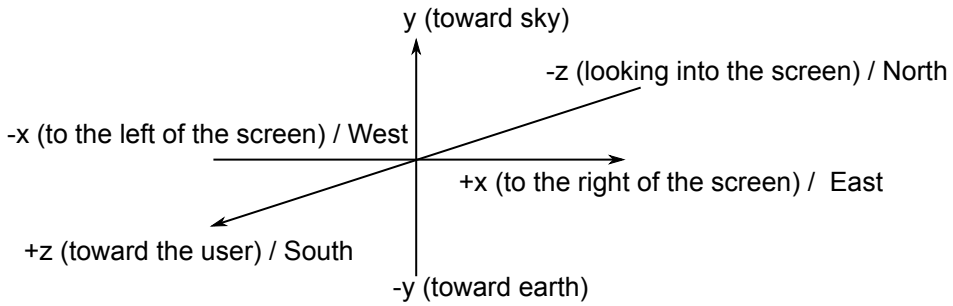
Coordinate Systems and Others



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.
...) { ... }
```

ARKit Coordinate System



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();
    }
}
```