Working with Metal—Overview

Session 603

Jeremy Sandmel
GPU Software

Metal

Dramatically reduced overhead

Unified graphics and compute

Precompiled shaders

Efficient multithreading

Designed for A7

Agenda

Background

API concepts

Shading language

Developer tools

Agenda

Background

API concepts

Shading language

Developer tools

10x more draw calls

About Draw Calls…

About Draw Calls…

Each draw call requires its own state vector

- Shaders, states, textures, render targets, etc.

About Draw Calls…

Each draw call requires its own state vector

- Shaders, states, textures, render targets, etc.

Changing state vectors can be expensive

- Translation to hardware commands

About Draw Calls…

Each draw call requires its own state vector

- Shaders, states, textures, render targets, etc.

Changing state vectors can be expensive for the CPU

- Translation to hardware commands

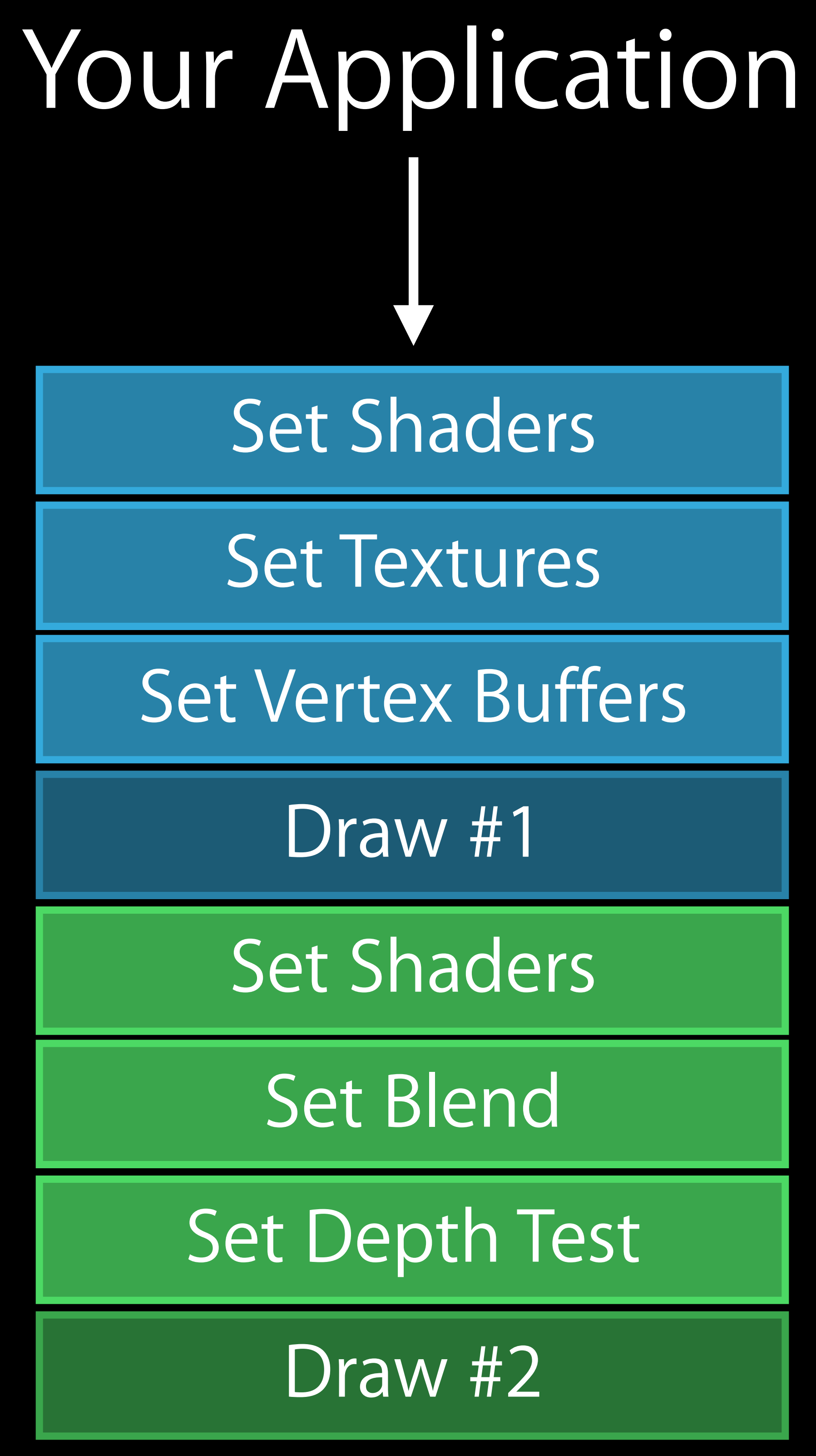
About Draw Calls…

Each draw call requires its own state vector

- Shaders, states, textures, render targets, etc.

Changing state vectors can be expensive for the CPU

- Translation to hardware commands



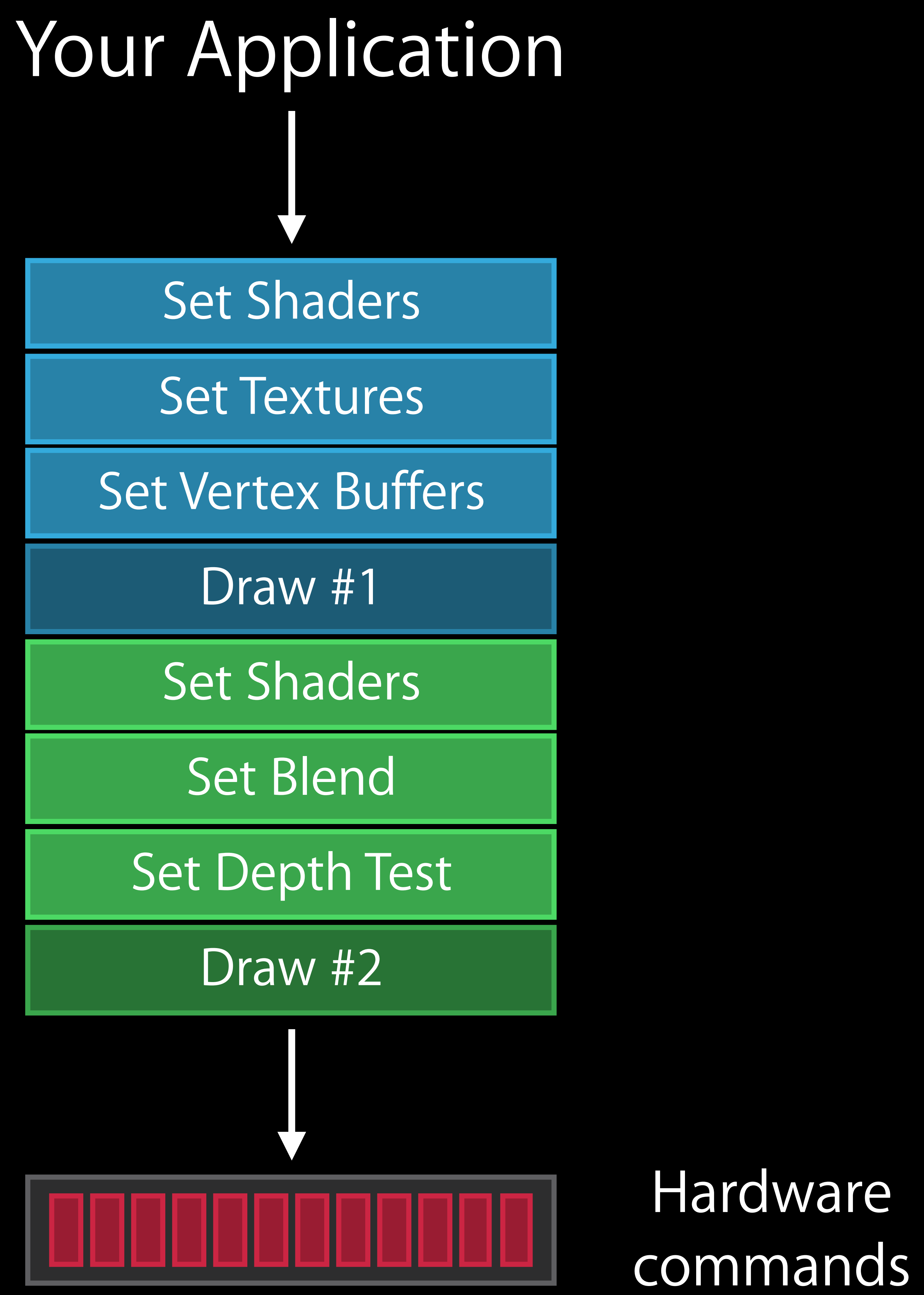
About Draw Calls…

Each draw call requires its own state vector

- Shaders, states, textures, render targets, etc.

Changing state vectors can be expensive for the CPU

- Translation to hardware commands



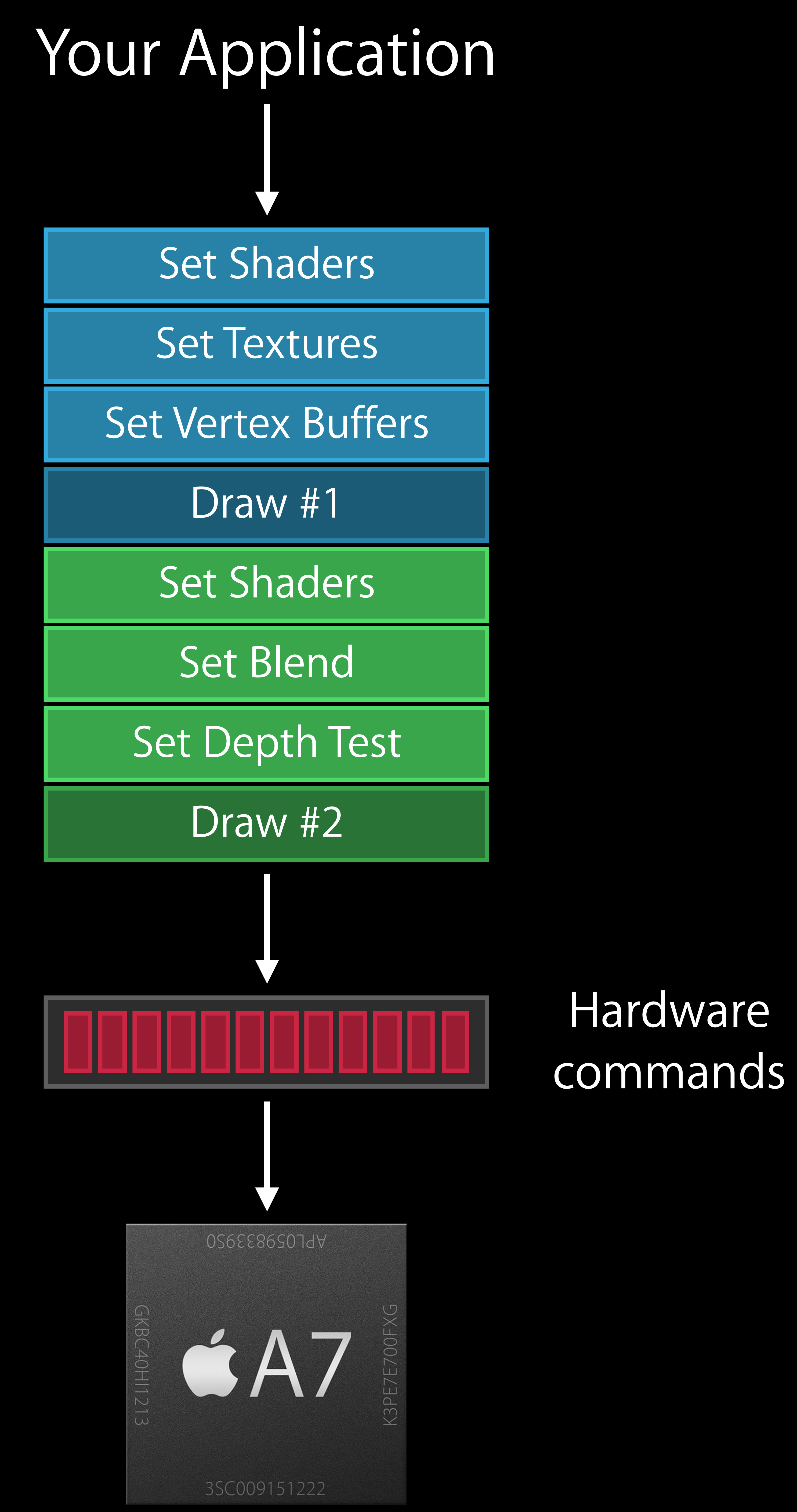
About Draw Calls…

Each draw call requires its own state vector

- Shaders, states, textures, render targets, etc.

Changing state vectors can be expensive for the CPU

- Translation to hardware commands



About Draw Calls…

Each draw call requires its own state vector

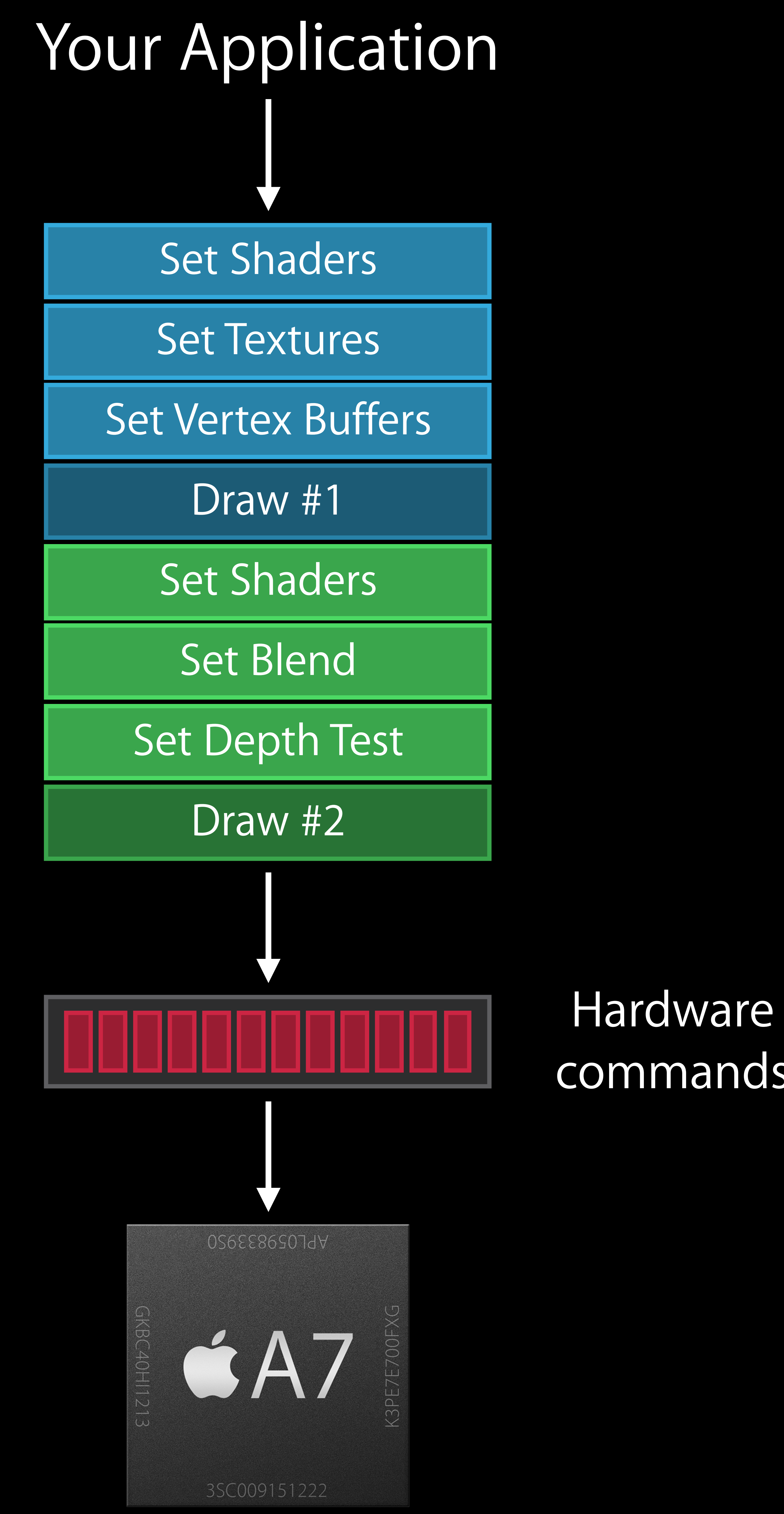
- Shaders, states, textures, render targets, etc.

Changing state vectors can be expensive for the CPU

- Translation to hardware commands

More draw calls per frame gives you

- More unique objects
- More visual variety
- More freedom for game artists and designers



Before Metal

Before Metal

Long history of GPU programming APIs

- Standards—OpenGL, OpenCL
- Domains—High level, low level, 2D, 3D
- Architectures—Platforms, devices, GPUs

Before Metal

Long history of GPU programming APIs

- Standards—OpenGL, OpenCL
- Domains—High level, low level, 2D, 3D
- Architectures—Platforms, devices, GPUs

Something was missing…

Deep Integration

Deep Integration

+

+

Deep Integration

Deep Integration

What if we took the same approach for GPU programming?

Deep Integration

What if we took the same approach for GPU programming?

+

Deep Integration

What if we took the same approach for GPU programming?

Metal Design

Metal Design

Thinnest possible API

Metal Design

Thinnest possible API

Modern GPU features

Metal Design

Thinnest possible API

Modern GPU features

Do expensive tasks less often

Metal Design

Thinnest possible API

Modern GPU features

Do expensive tasks less often

Predictable performance

Metal Design

Thinnest possible API

Modern GPU features

Do expensive tasks less often

Predictable performance

Explicit command submission

Metal Design

Thinnest possible API

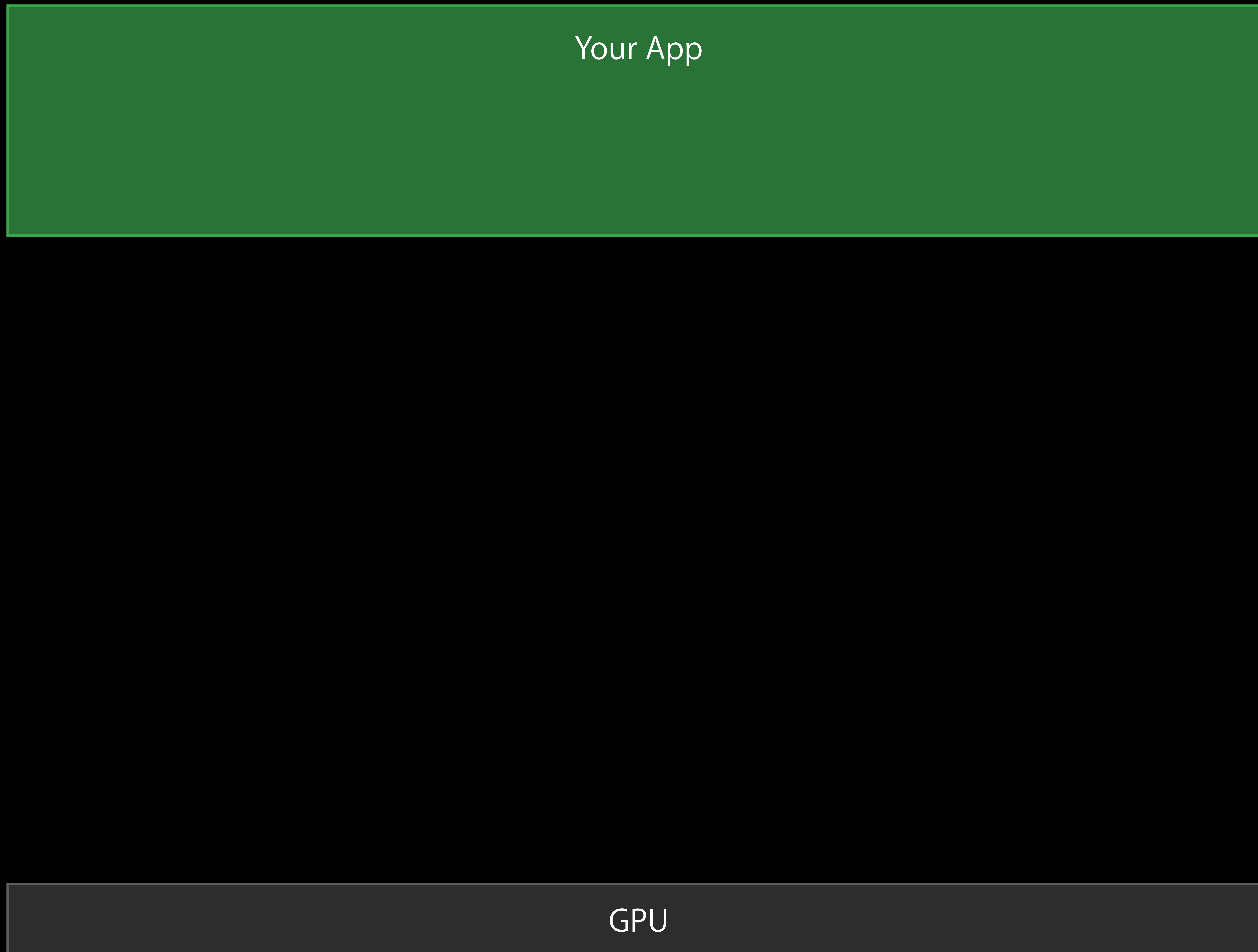
Modern GPU features

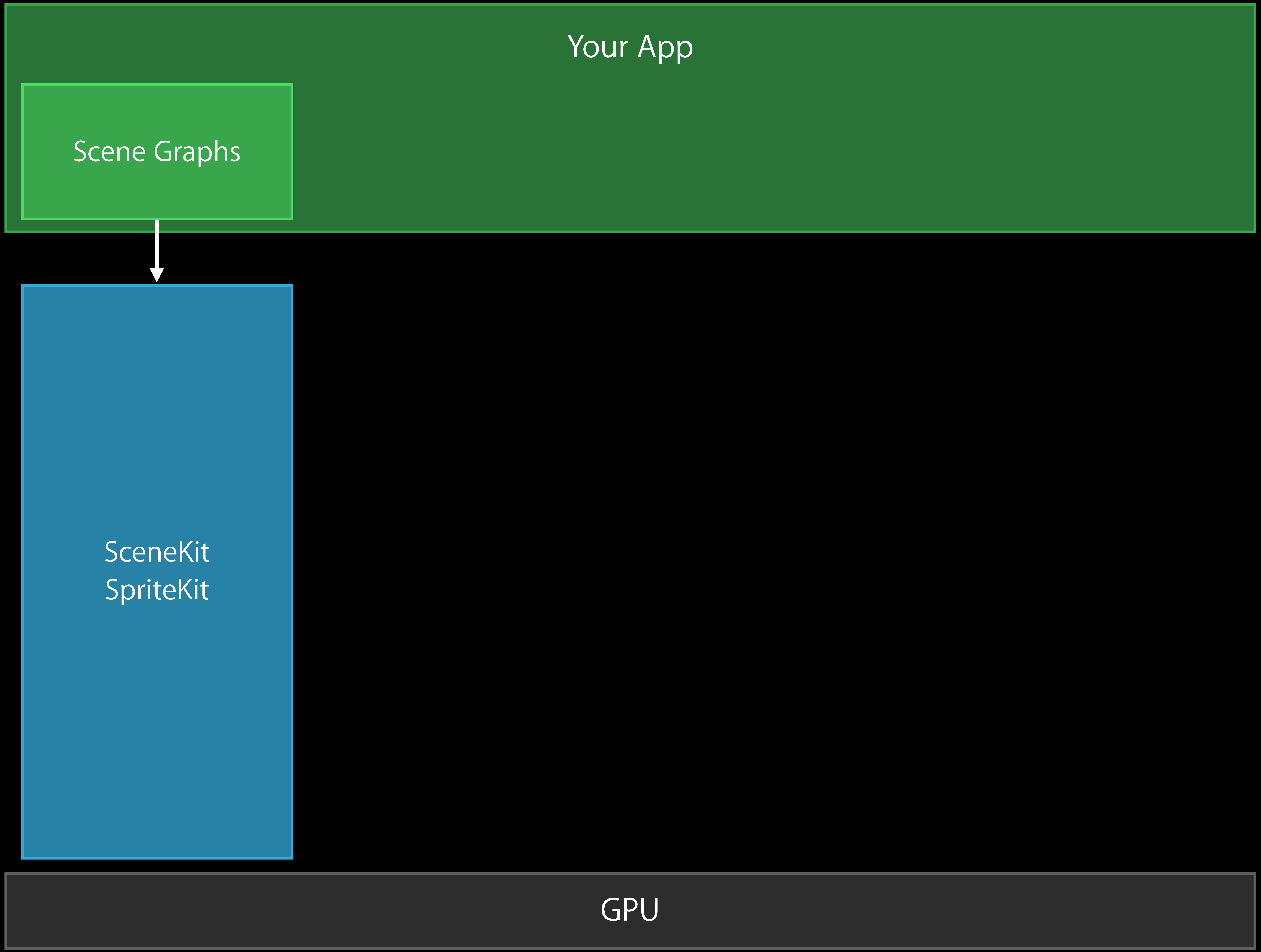
Do expensive tasks less often

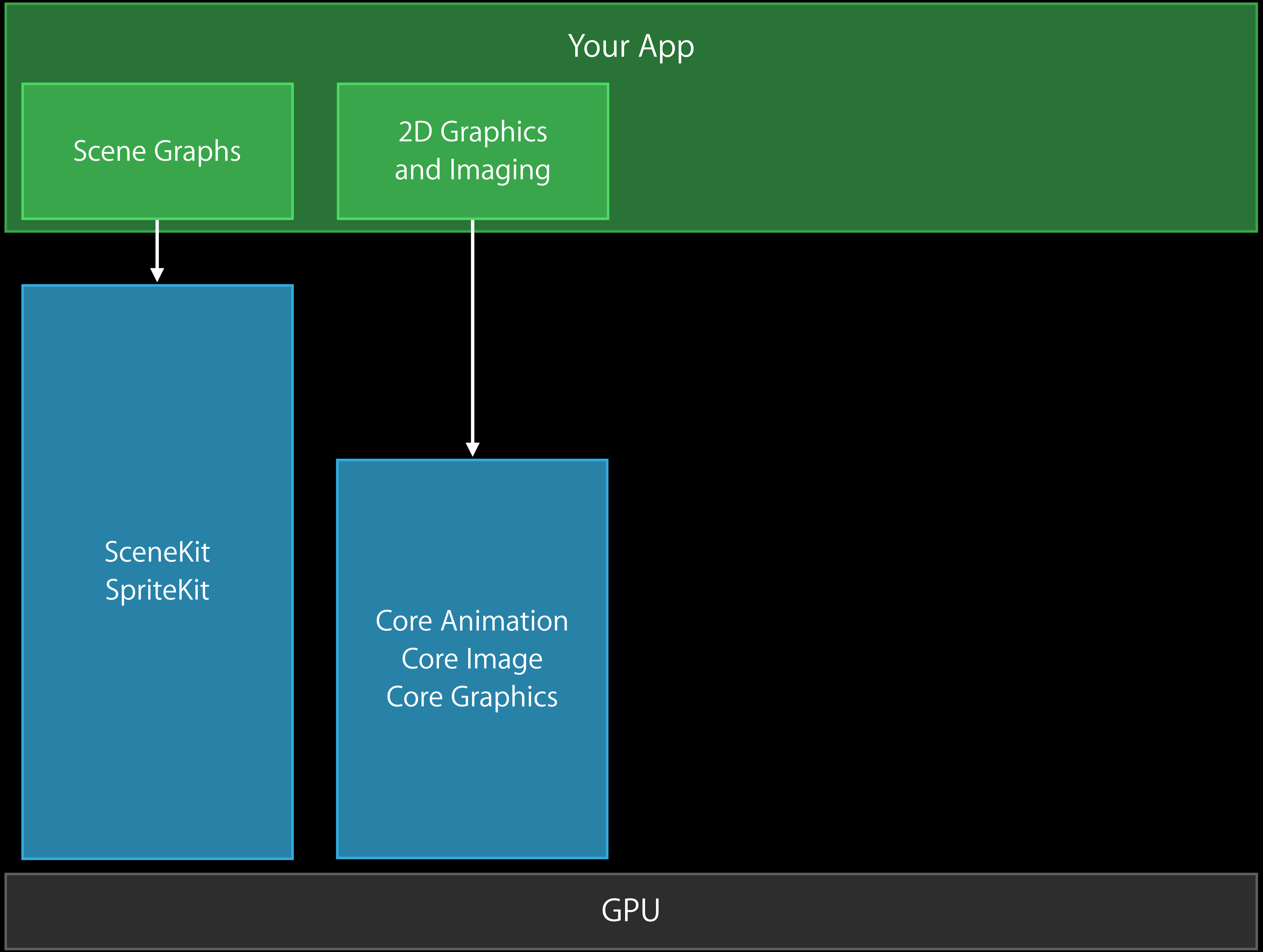
Predictable performance

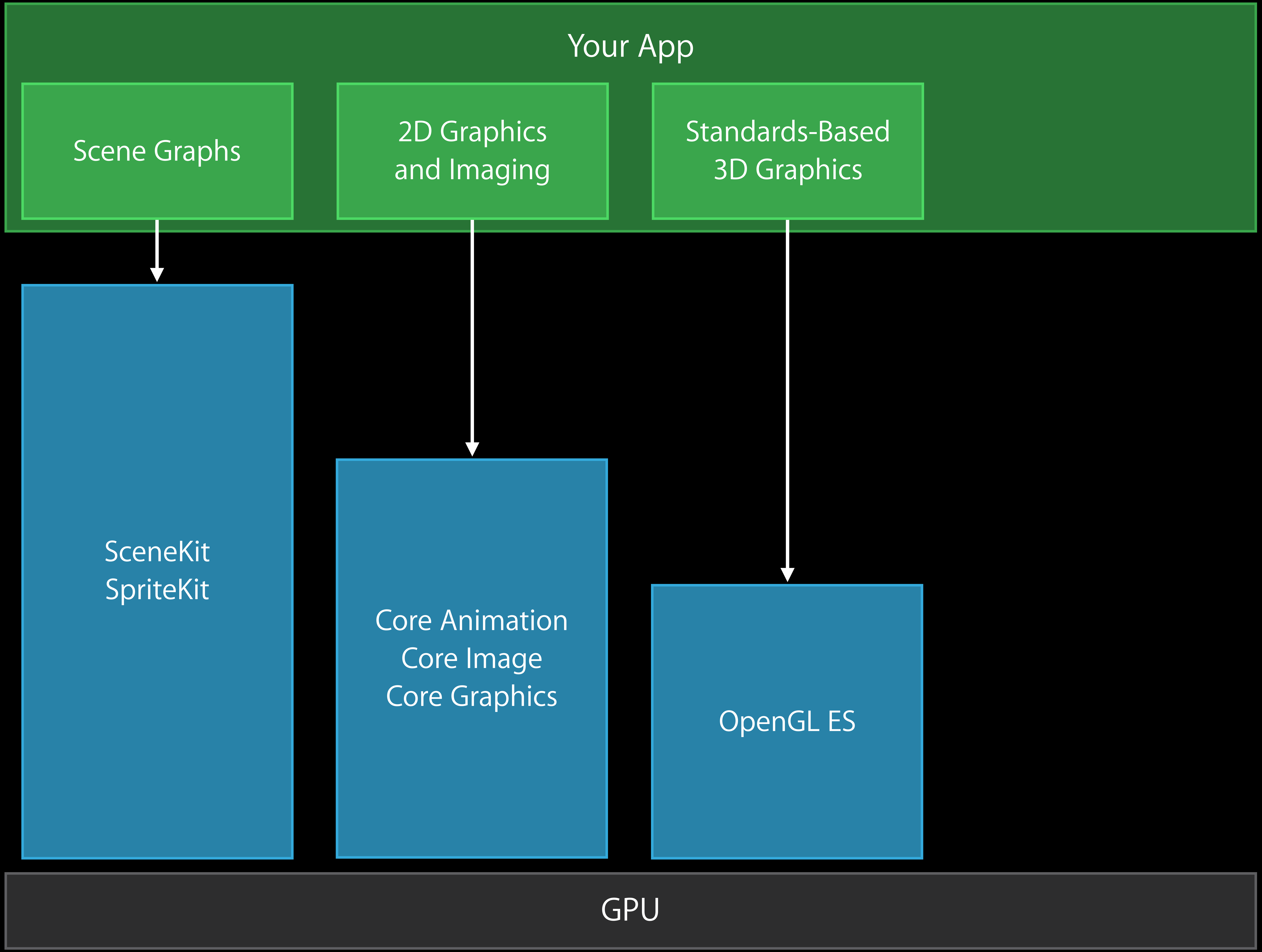
Explicit command submission

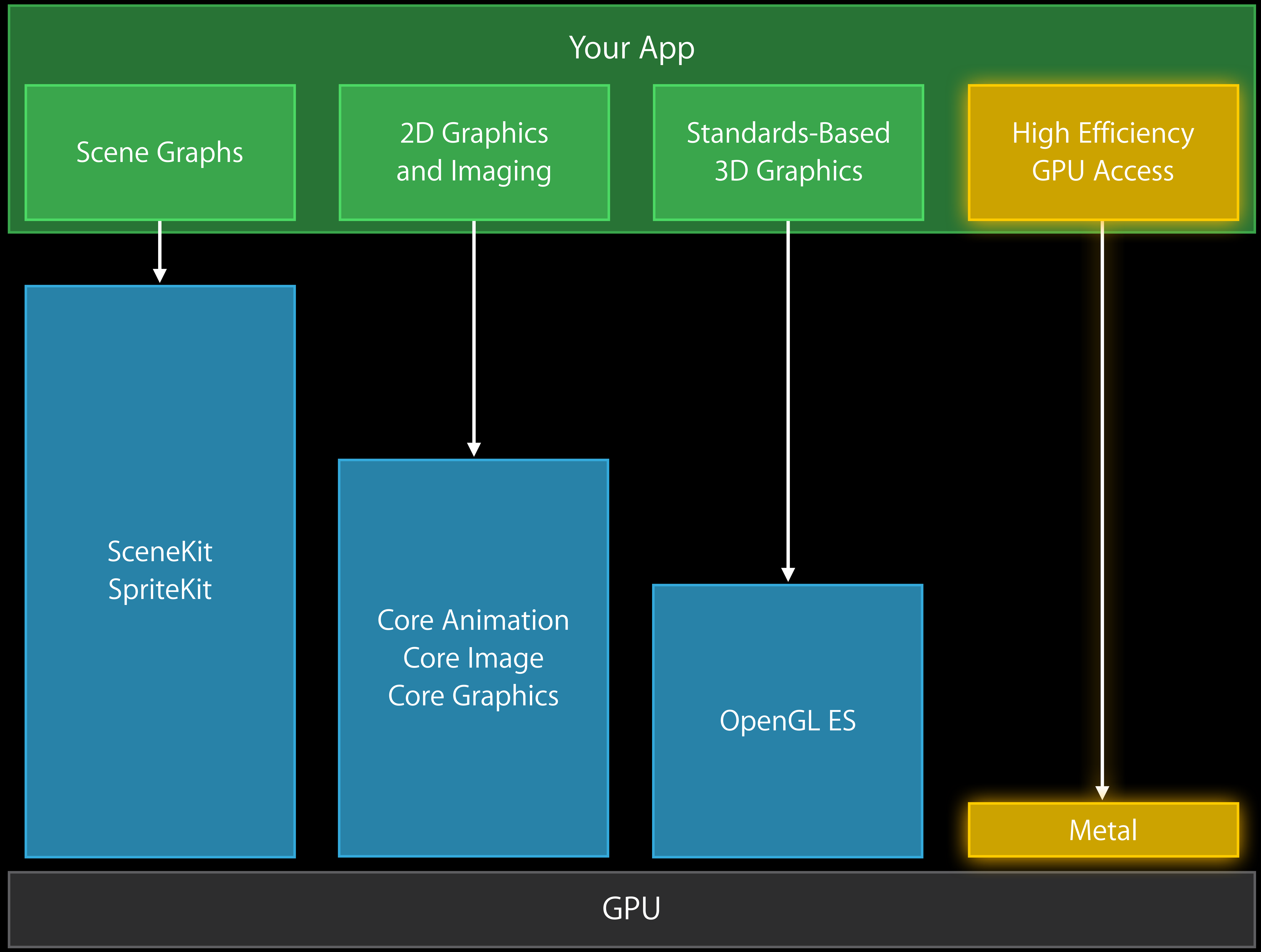
Optimized for CPU behavior


Your App
GPU


Your App
Scene Graphs
SceneKit
SpriteKit
GPU


Your App
Scene Graphs
2D Graphics
and Imaging
SceneKit
SpriteKit
Core Animation
Core Image
Core Graphics
GPU


Your App
Scene Graphs
2D Graphics and Imaging
Standards-Based 3D Graphics
SceneKit
SpriteKit
Core Animation
Core Image
Core Graphics
OpenGL ES
GPU


Your App
Scene Graphs
2D Graphics and Imaging
Standards-Based 3D Graphics
High Efficiency GPU Access
SceneKit
SpriteKit
Core Animation
Core Image
Core Graphics
OpenGL ES
Metal
GPU

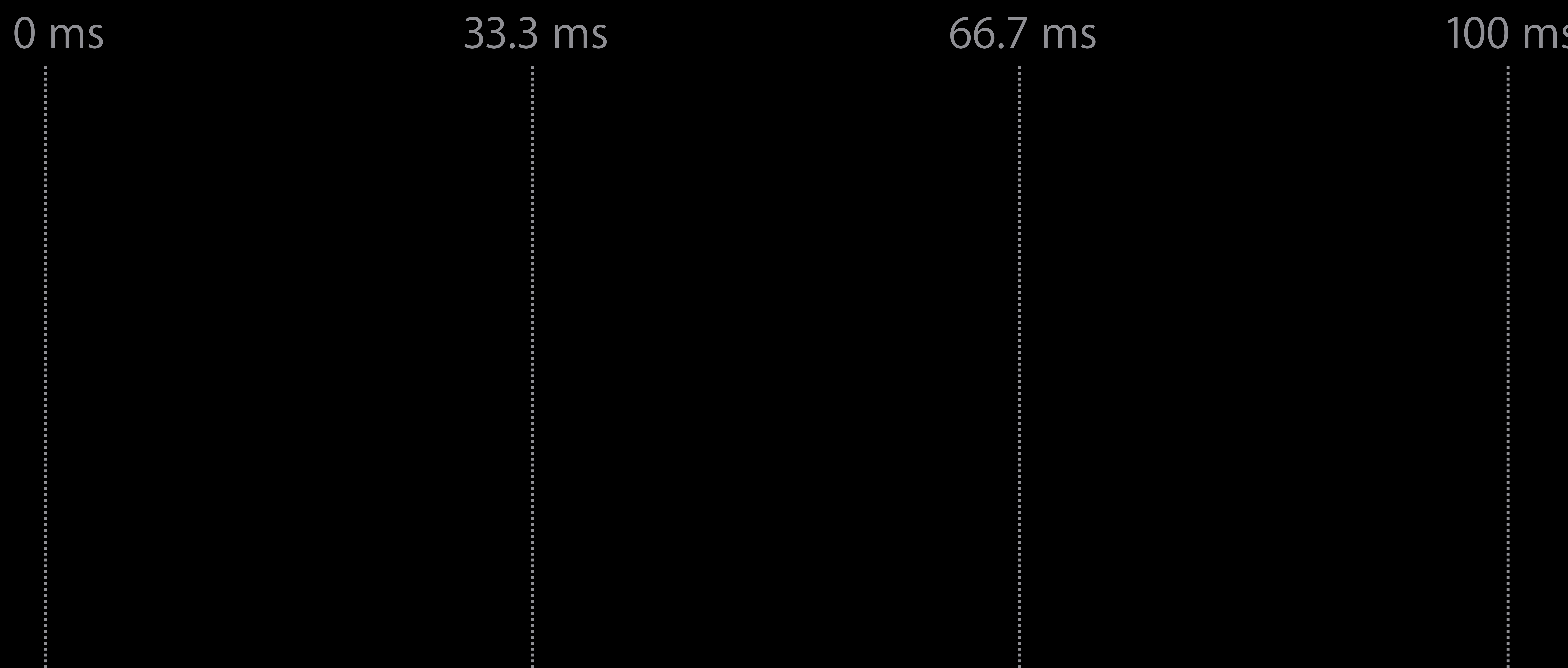
So how did we do this?

Frame Times

Many games target frame rate of 30 FPS (33.3 milliseconds/frame)

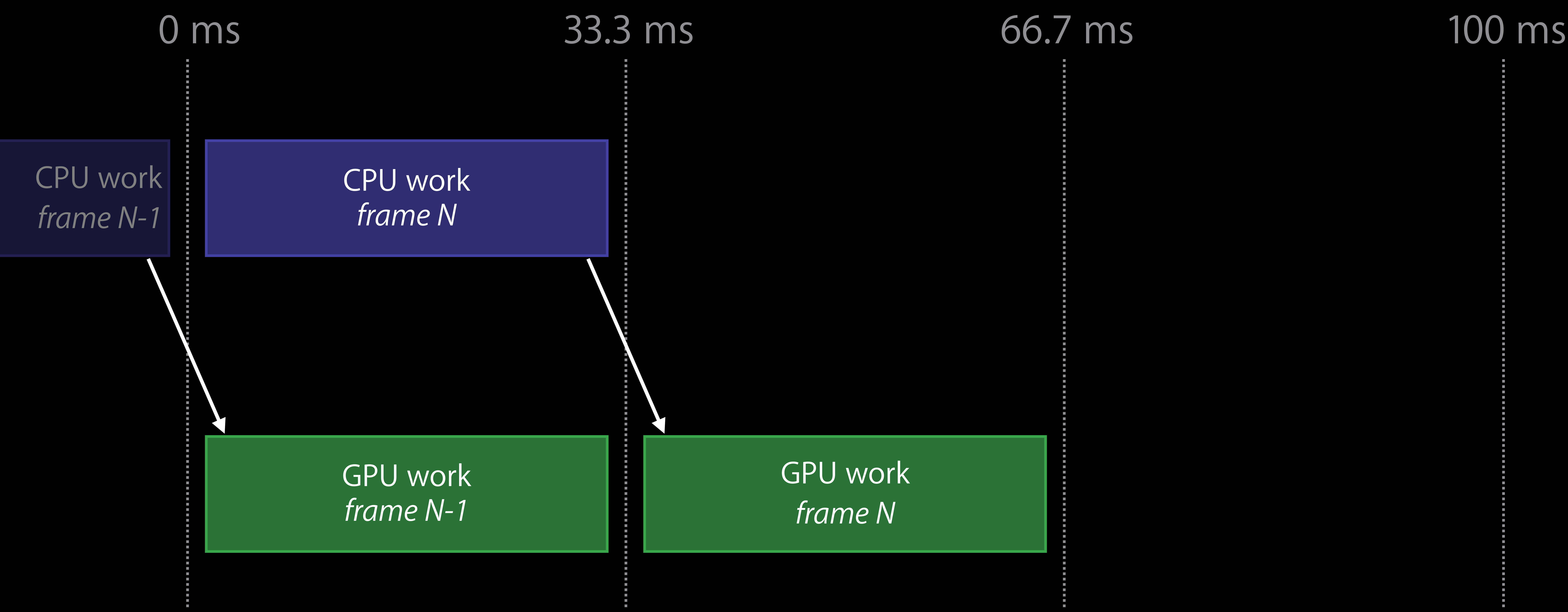
Frame Times

Many games target frame rate of 30 FPS (33.3 milliseconds/frame)



Frame Times

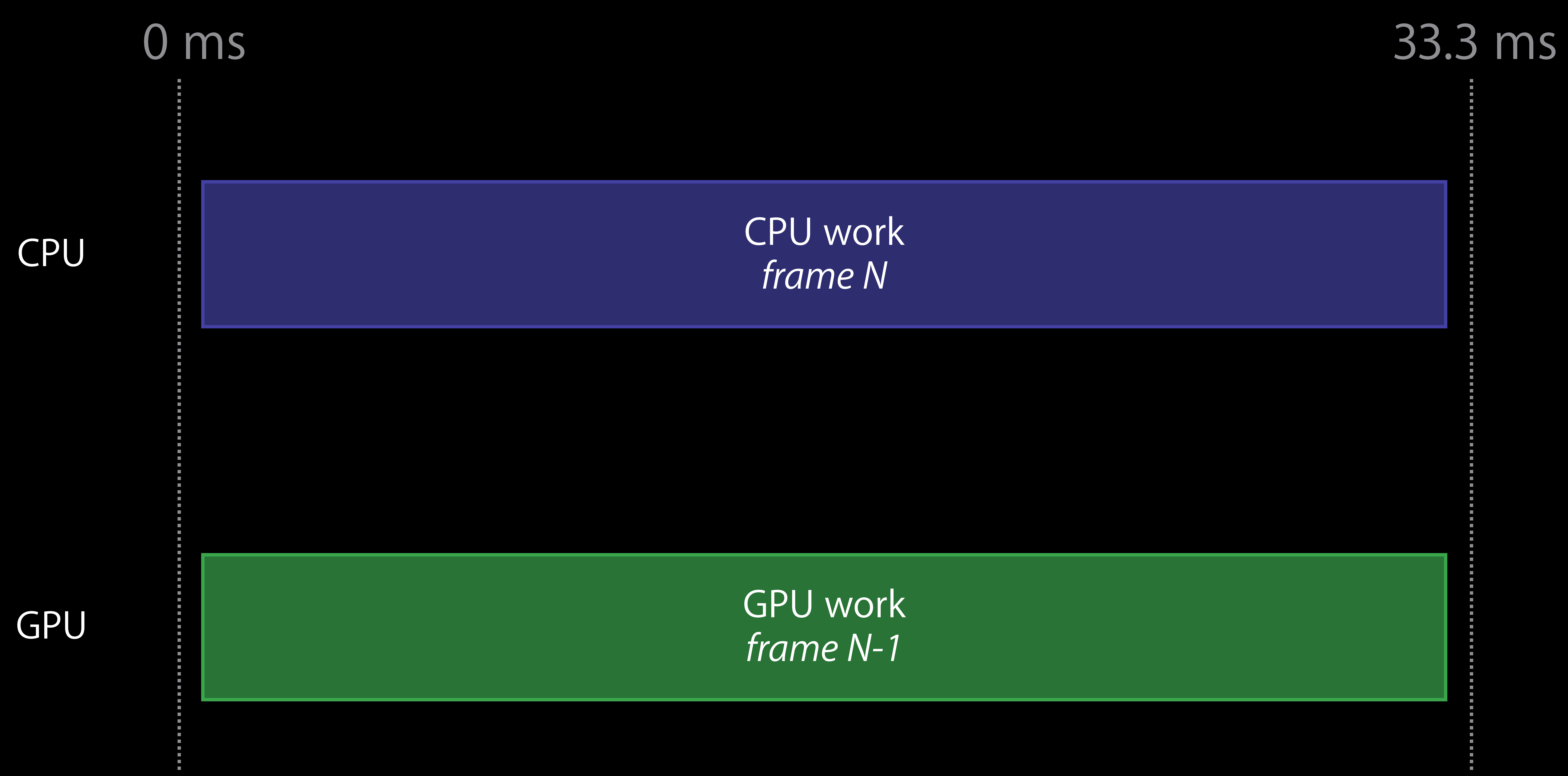
Many games target frame rate of 30 FPS (33.3 milliseconds/frame)



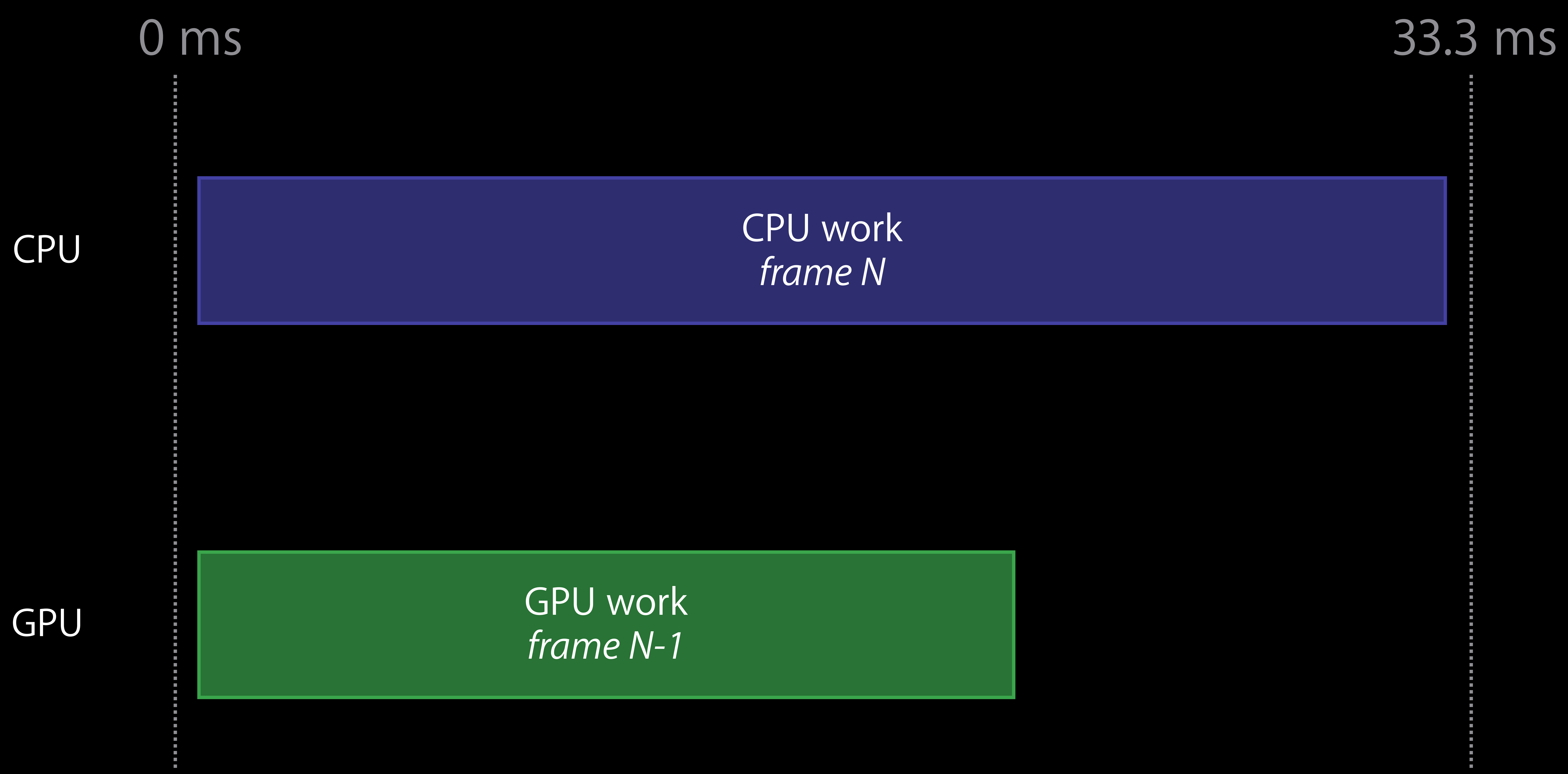
Frame Times

Many games target frame rate of 30 FPS (33.3 milliseconds/frame)

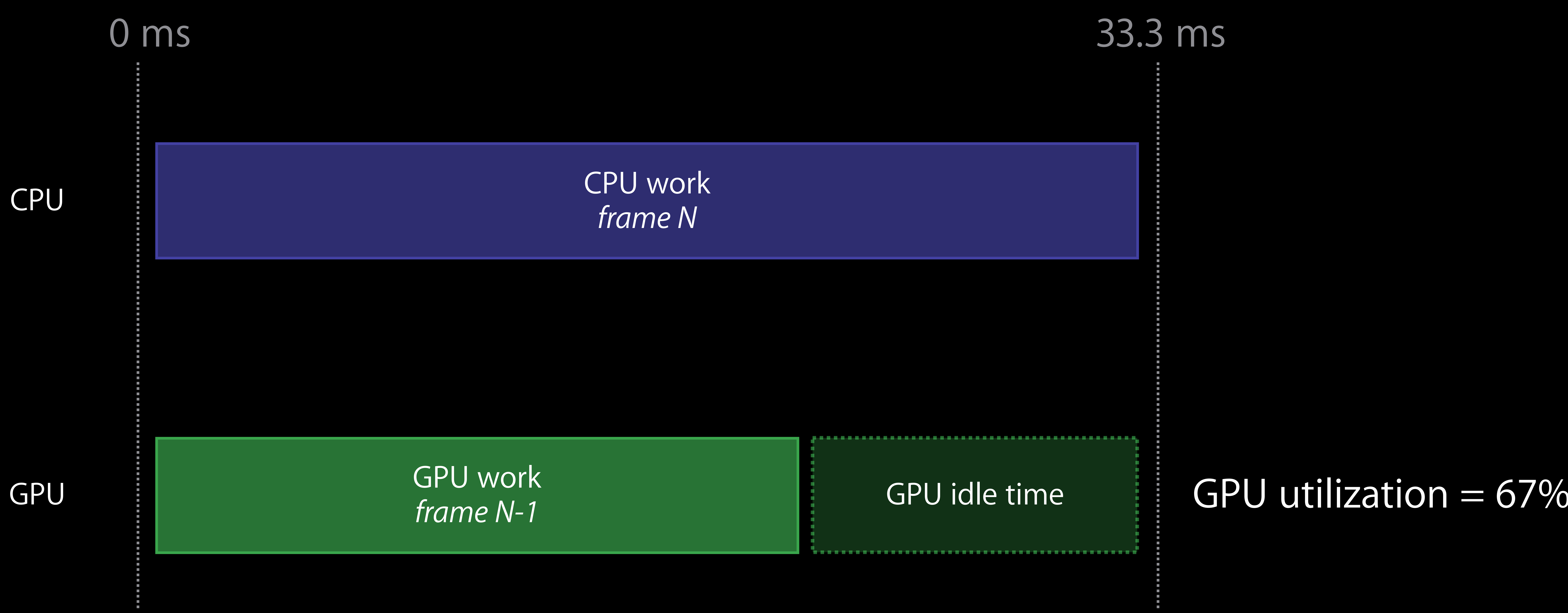
One “Balanced” Frame



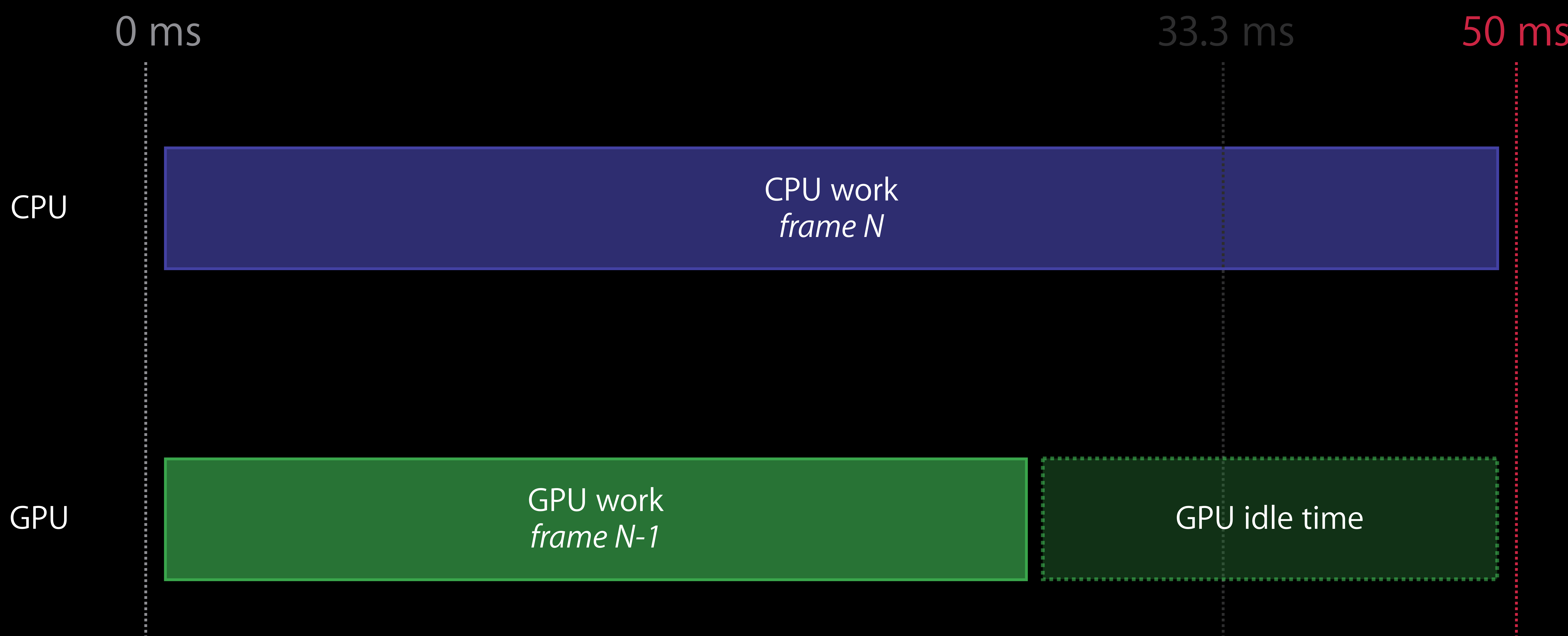
CPU Can Take More Time Than GPU



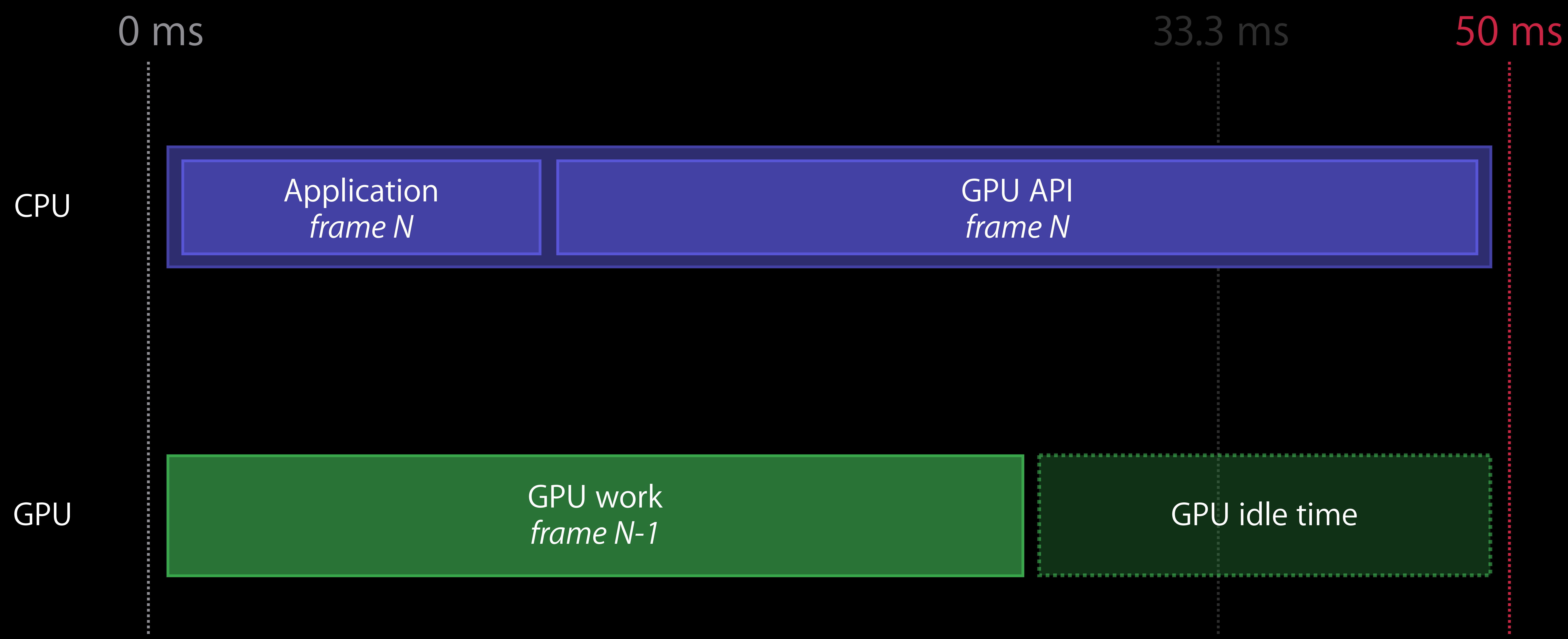
CPU Can Take More Time Than GPU



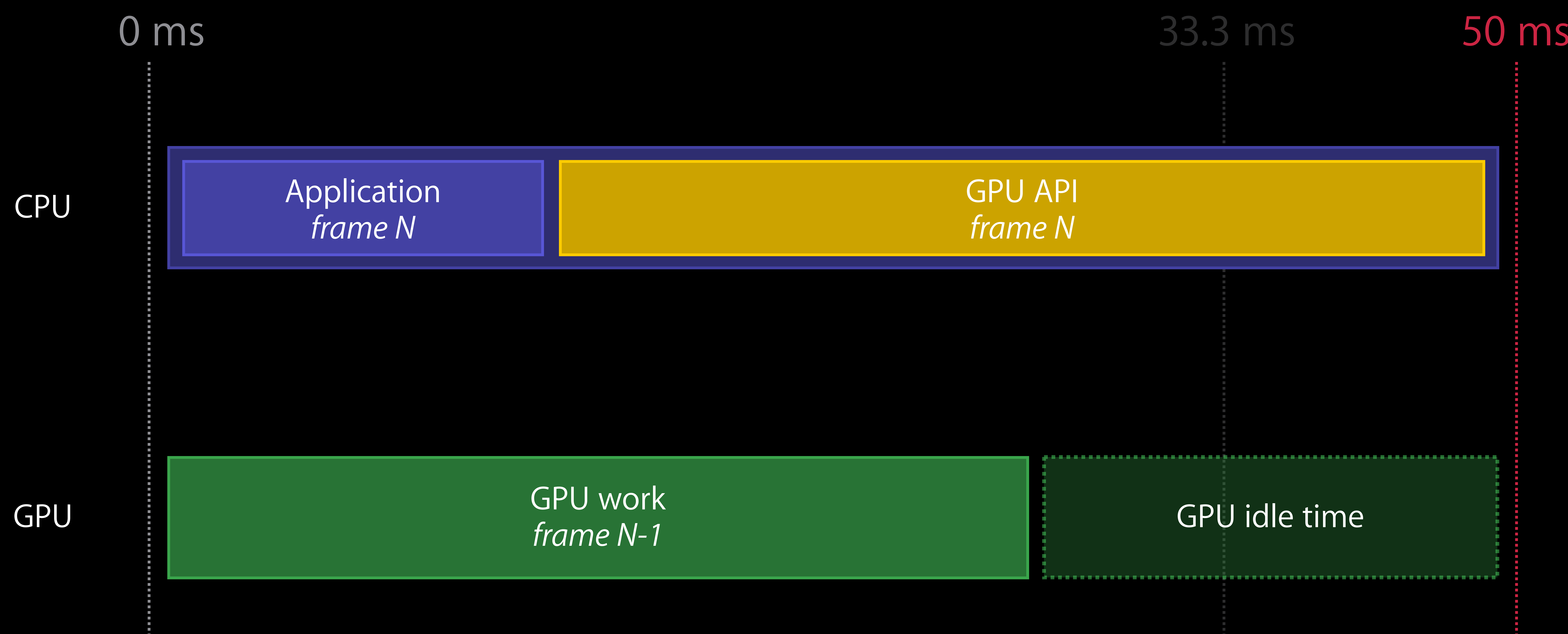
More GPU Work Requires More CPU Work



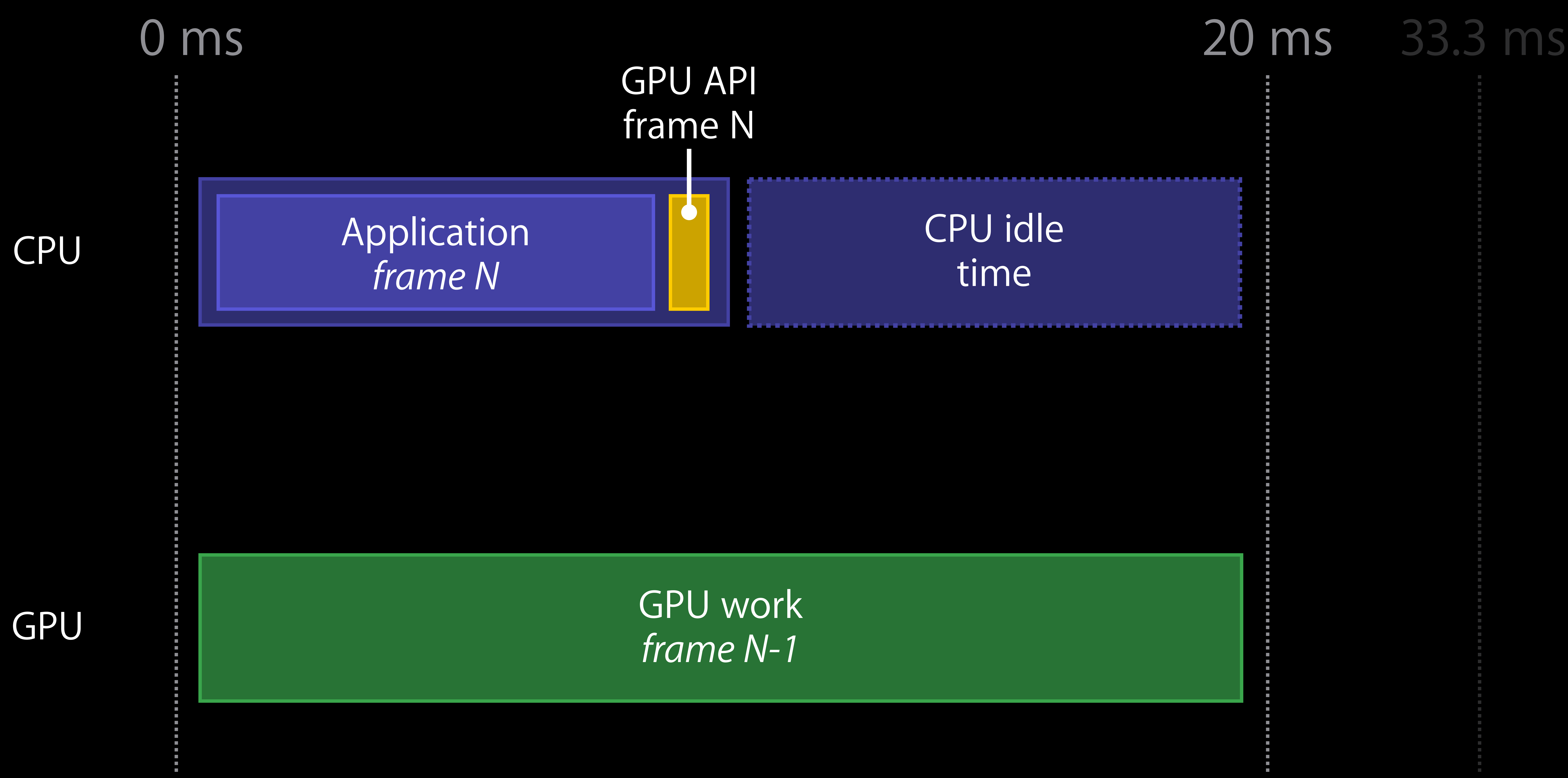
CPU Time Includes Application and GPU API



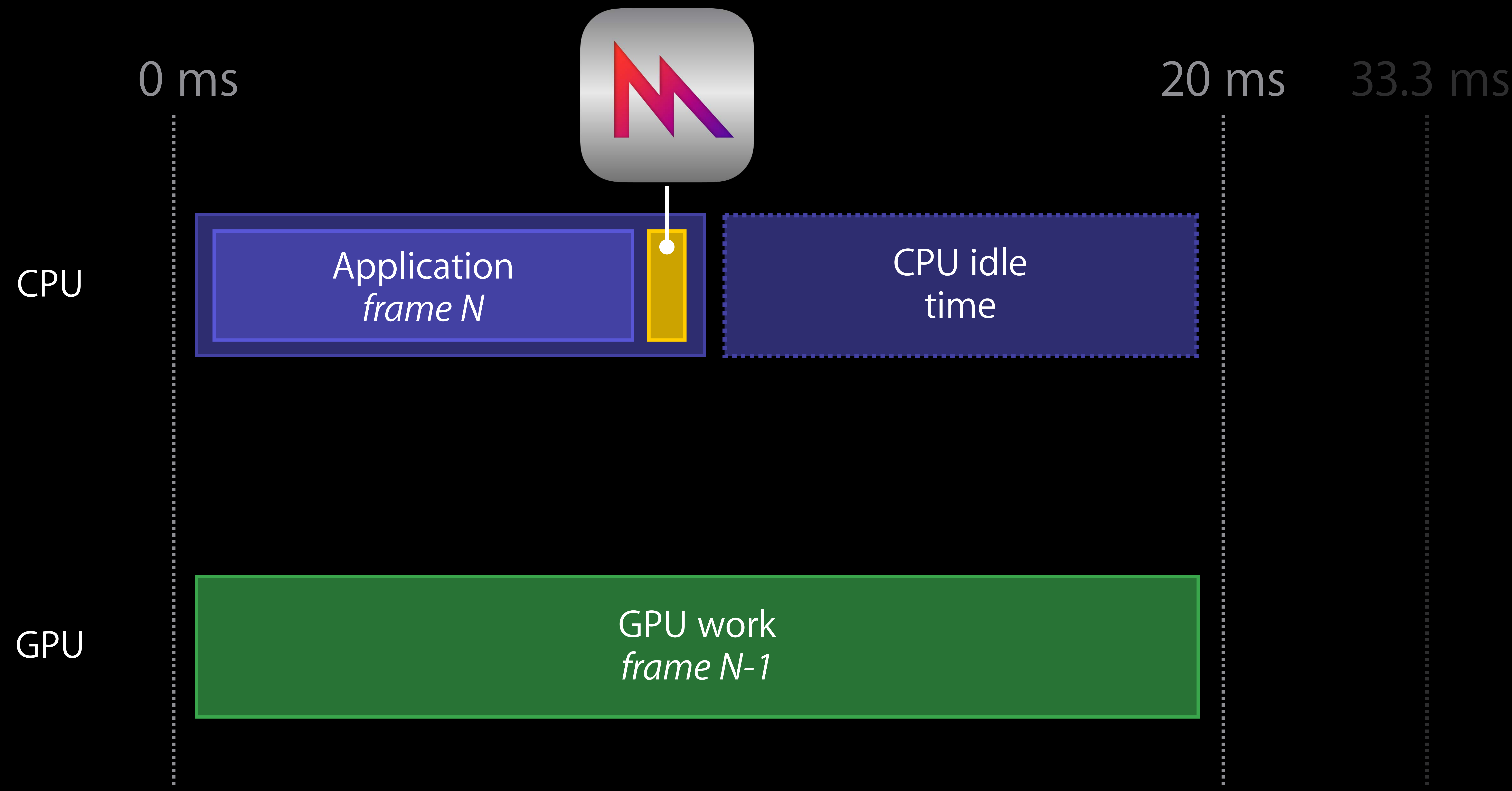
Targeting CPU Time Spent in GPU API



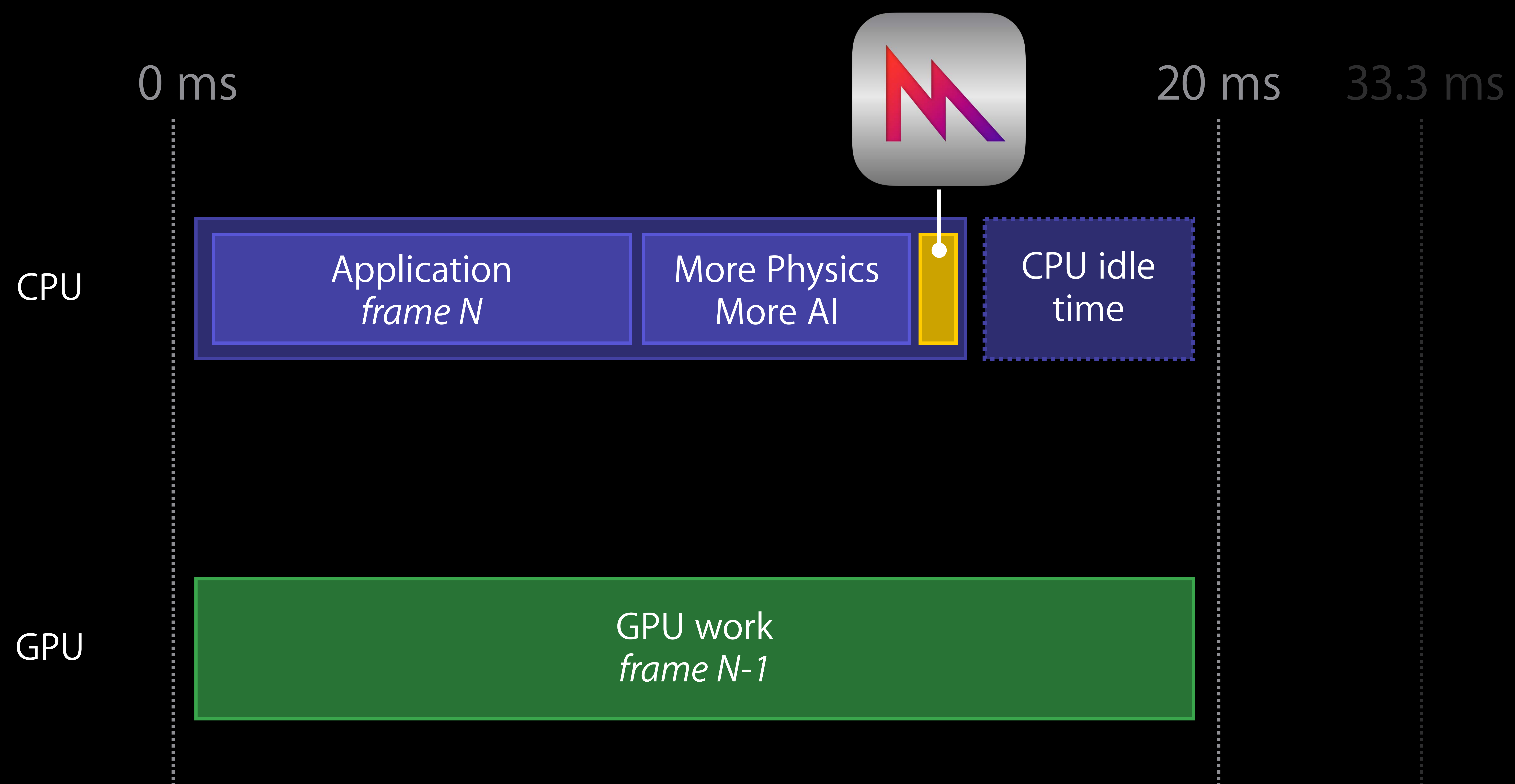
Metal Dramatically Reduces GPU API Time



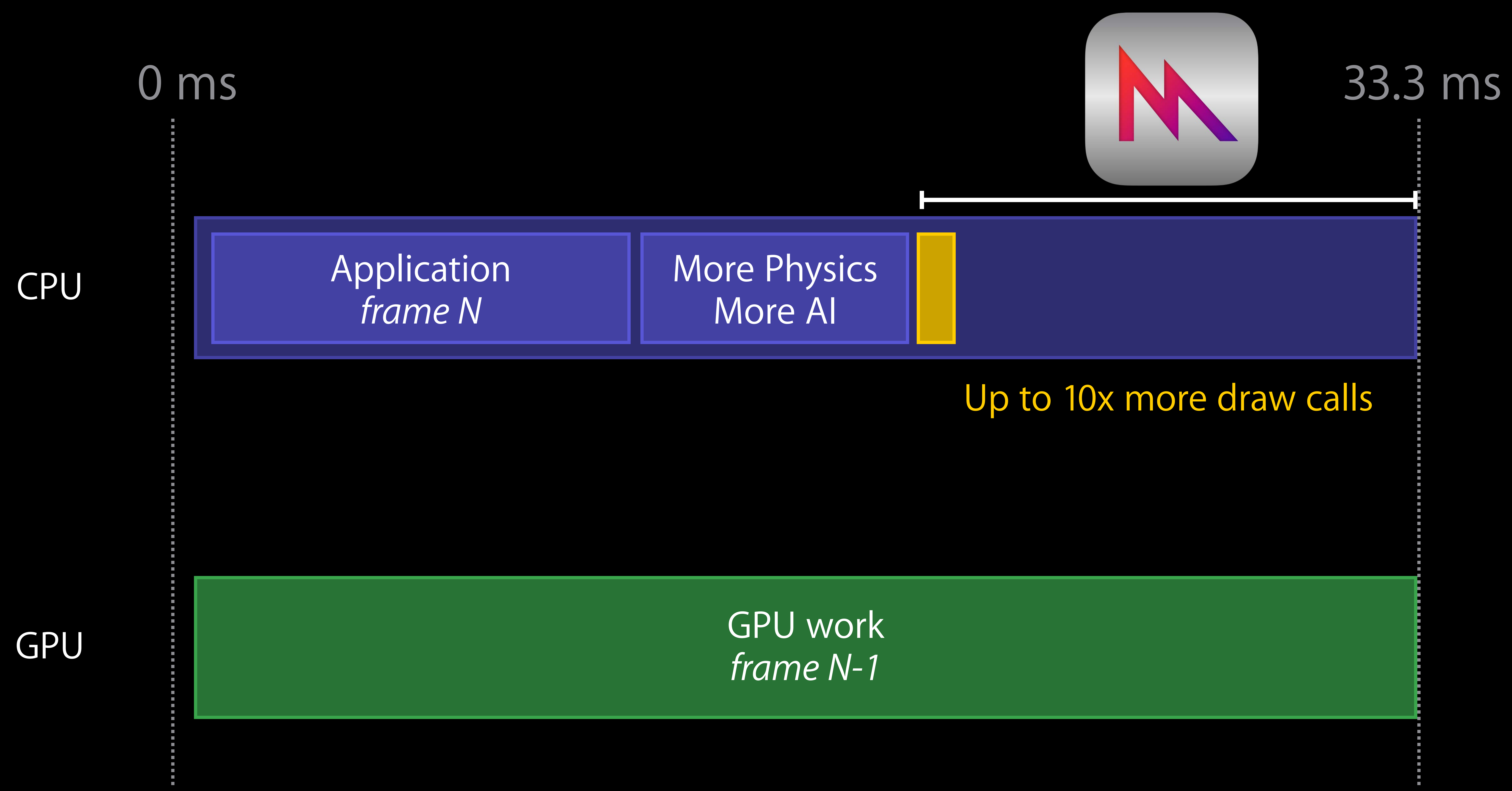
Metal Dramatically Reduces GPU API Time



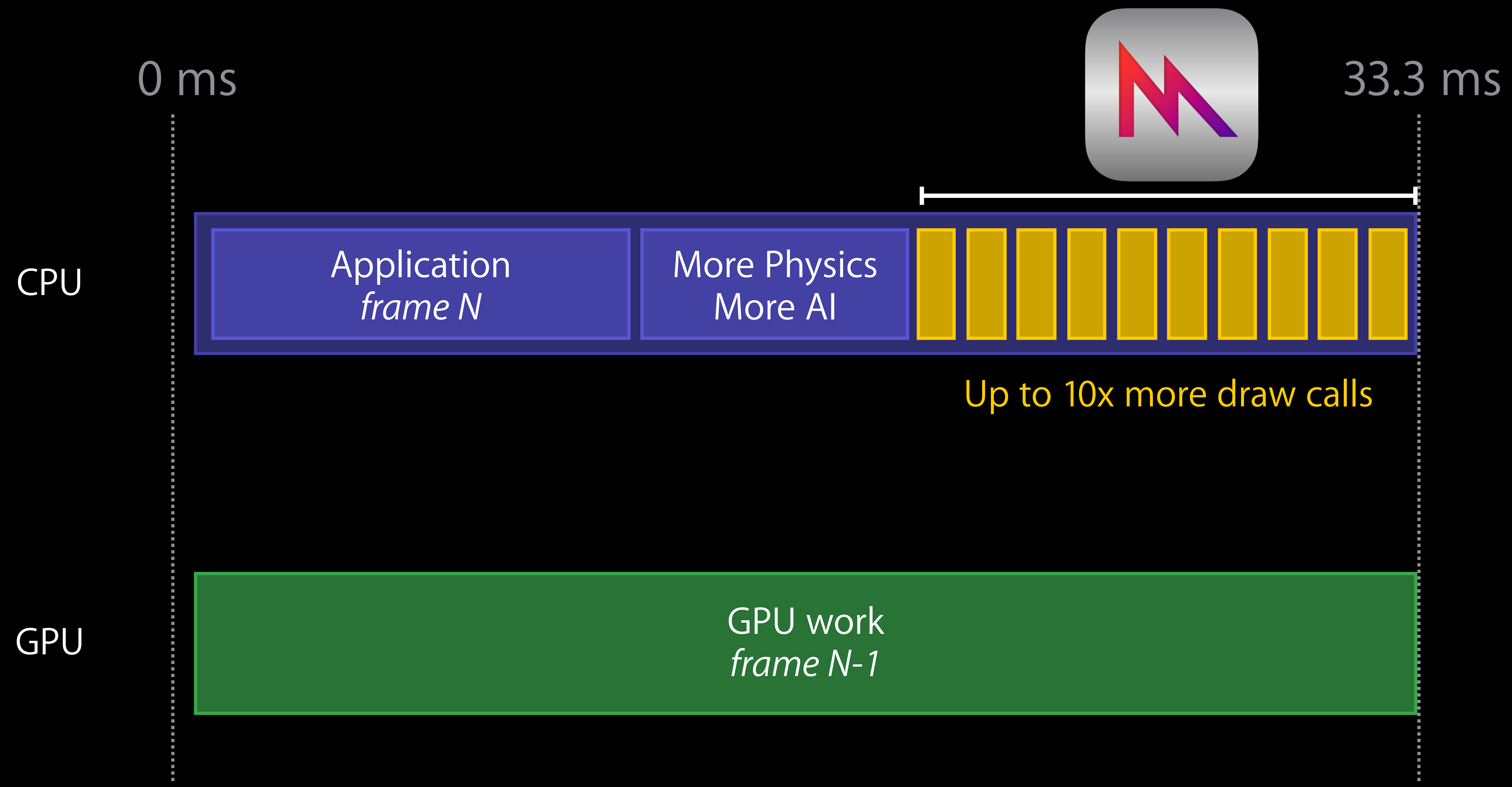
Use CPU Time to Improve Your Game



Use CPU Time to Draw More Objects



Use CPU Time to Draw More Objects



Why Is GPU Programming Expensive?

Why Is GPU Programming Expensive?

State validation

- Confirming API usage is valid
- Encoding API state to hardware state

Why Is GPU Programming Expensive?

State validation

- Confirming API usage is valid
- Encoding API state to hardware state

Shader compilation

- Run-time generation of shader machine code
- Interactions between state and shaders

Why Is GPU Programming Expensive?

State validation

- Confirming API usage is valid
- Encoding API state to hardware state

Shader compilation

- Run-time generation of shader machine code
- Interactions between state and shaders

Sending work to GPU

- Managing resource residency
- Batching commands

Do Expensive Tasks Less Often

When	Frequency
Application build	
Content loading	
Draw time	

Do Expensive Tasks Less Often

When	Frequency	
Application build	"Never"	
Content loading		
Draw time		

Do Expensive Tasks Less Often

When	Frequency	
Application build	"Never"	
Content loading	Rare	
Draw time		

Do Expensive Tasks Less Often

When	Frequency	
Application build	“Never”	
Content loading	Rare	
Draw time	1000s of times per frame	

Do Expensive Tasks Less Often

When	Frequency	Before Metal
Application build	"Never"	
Content loading	Rare	
Draw time	1000s of times per frame	Shader compilation State validation Start work on GPU

Do Expensive Tasks Less Often

When	Frequency	Before Metal	After Metal
Application build	"Never"		Shader compilation
Content loading	Rare		State validation
Draw time	1000s of times per frame	Shader compilation State validation Start work on GPU	Start work on GPU

Agenda

- Background
- API concepts
- Shading language
- Developer tools

Metal Objects

Objects	Purpose

Metal Objects

Objects	Purpose
Device	The GPU

Metal Objects

Objects	Purpose
Device	The GPU
Command Queue	Serial sequence of command buffers

Metal Objects

Objects	Purpose
Device	The GPU
Command Queue	Serial sequence of command buffers
Command Buffer	Contains GPU hardware commands

Metal Objects

Objects	Purpose
Device	The GPU
Command Queue	Serial sequence of command buffers
Command Buffer	Contains GPU hardware commands
Command Encoder	Translates API commands to GPU hardware commands

Metal Objects

Objects	Purpose
Device	The GPU
Command Queue	Serial sequence of command buffers
Command Buffer	Contains GPU hardware commands
Command Encoder	Translates API commands to GPU hardware commands
State	Framebuffer configuration, blend, depth, samplers, etc.

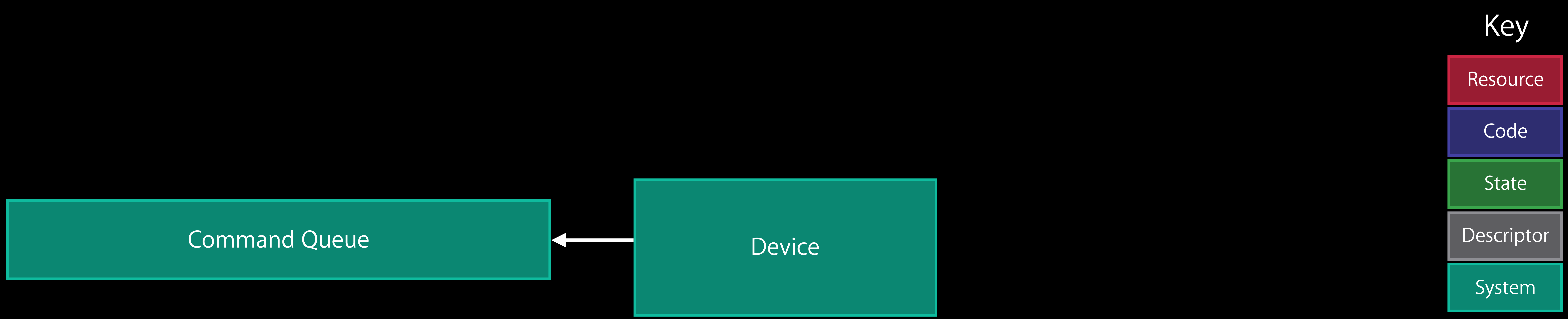
Metal Objects

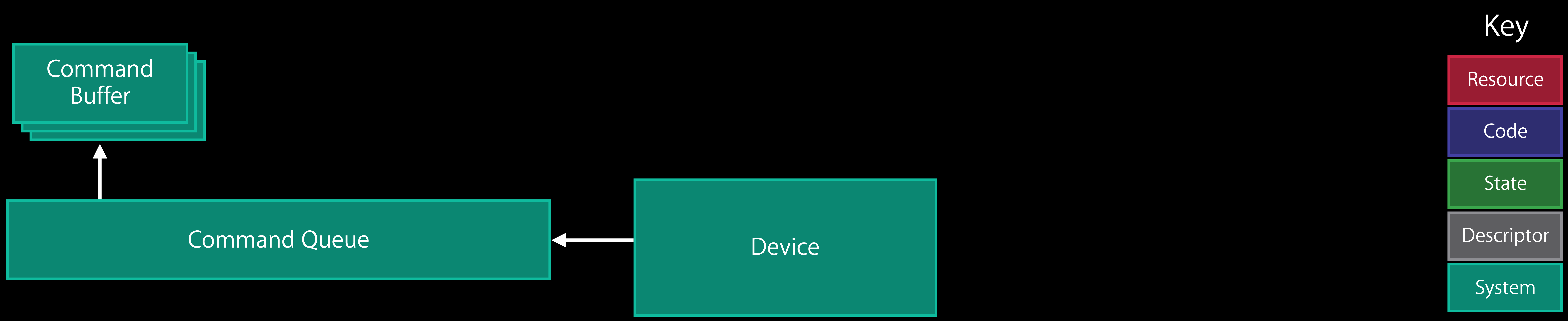
Objects	Purpose
Device	The GPU
Command Queue	Serial sequence of command buffers
Command Buffer	Contains GPU hardware commands
Command Encoder	Translates API commands to GPU hardware commands
State	Framebuffer configuration, blend, depth, samplers, etc.
Code	Shaders

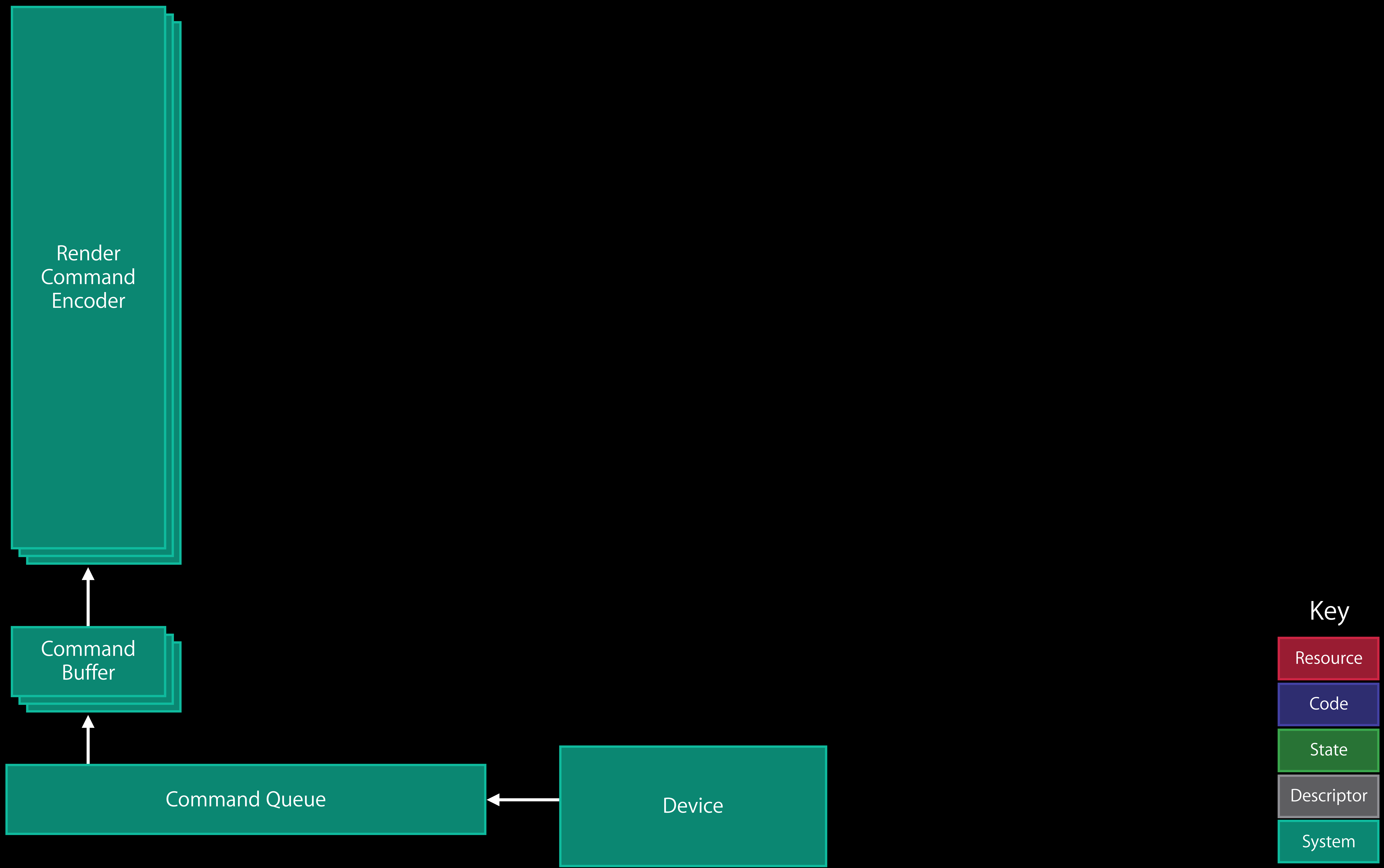
Metal Objects

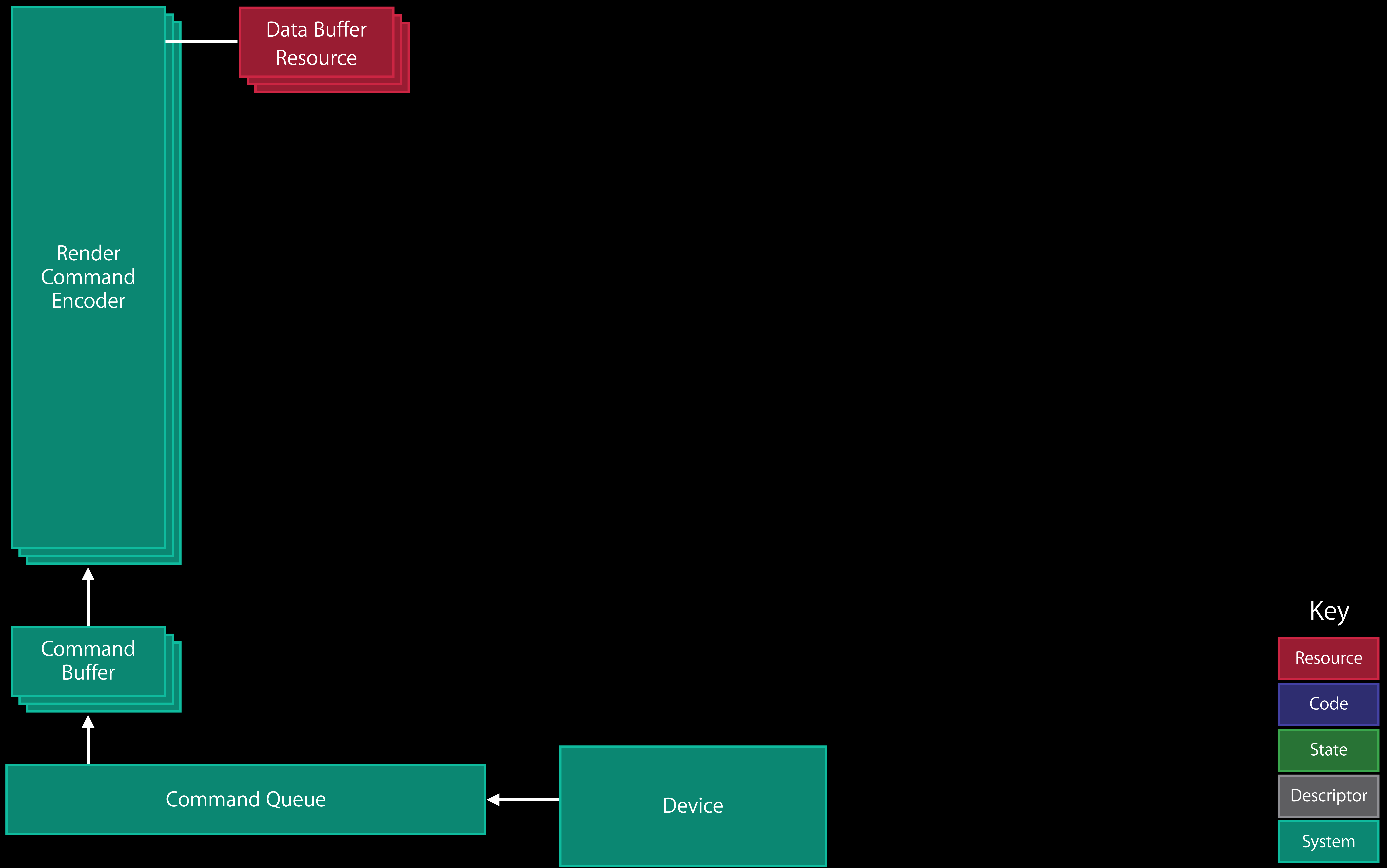
Objects	Purpose
Device	The GPU
Command Queue	Serial sequence of command buffers
Command Buffer	Contains GPU hardware commands
Command Encoder	Translates API commands to GPU hardware commands
State	Framebuffer configuration, blend, depth, samplers, etc.
Code	Shaders
Resources	Textures and Data Buffers (vertices, constants, etc.)

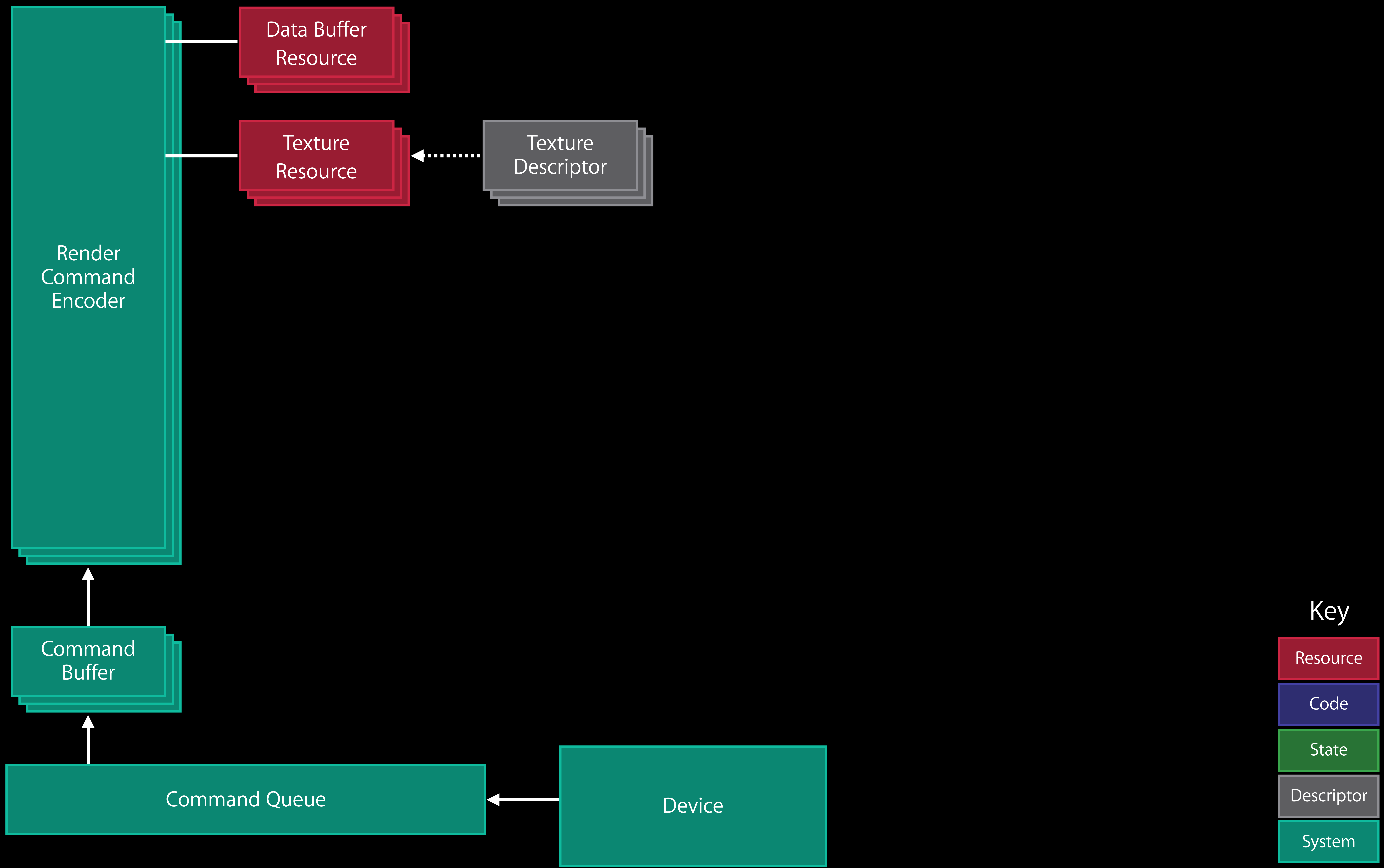

Device
Key
Resource
Code
State
Descriptor
System

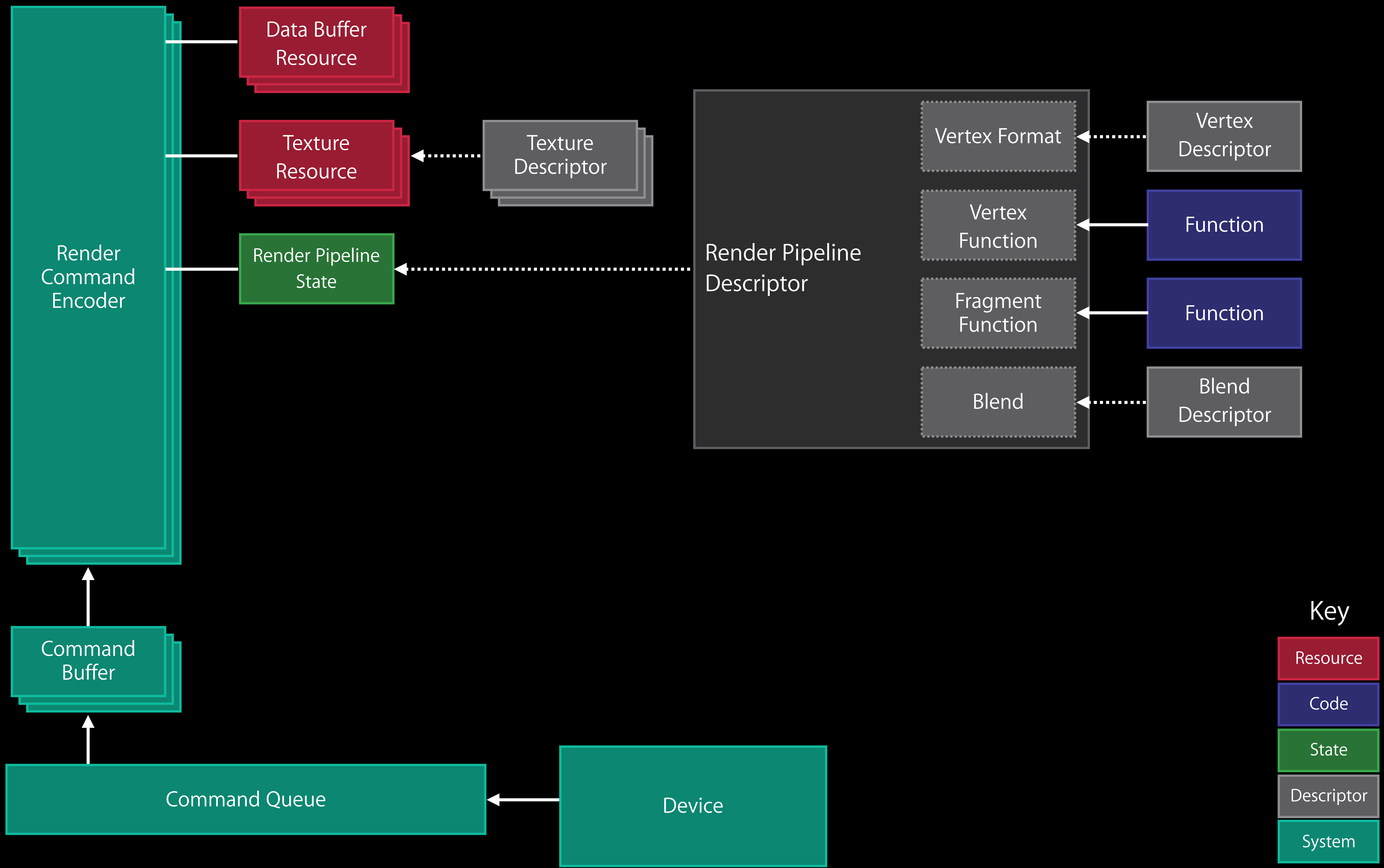

Command Queue
Device
Key
Resource
Code
State
Descriptor
System


Command
Buffer
Command Queue
Device
Key
Resource
Code
State
Descriptor
System

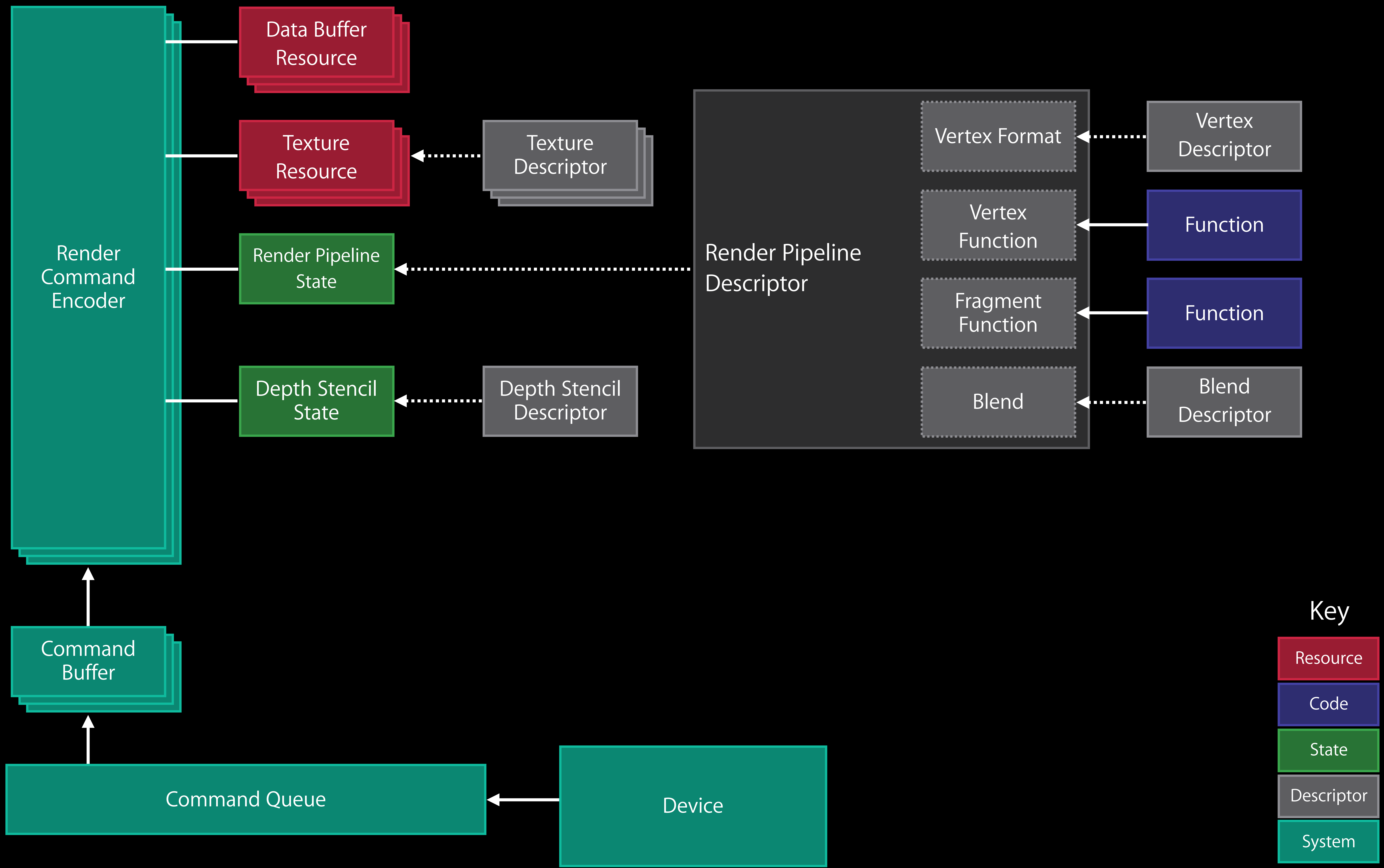

Render Command Encoder
Command Buffer
Command Queue
Device
Key
Resource
Code
State
Descriptor
System

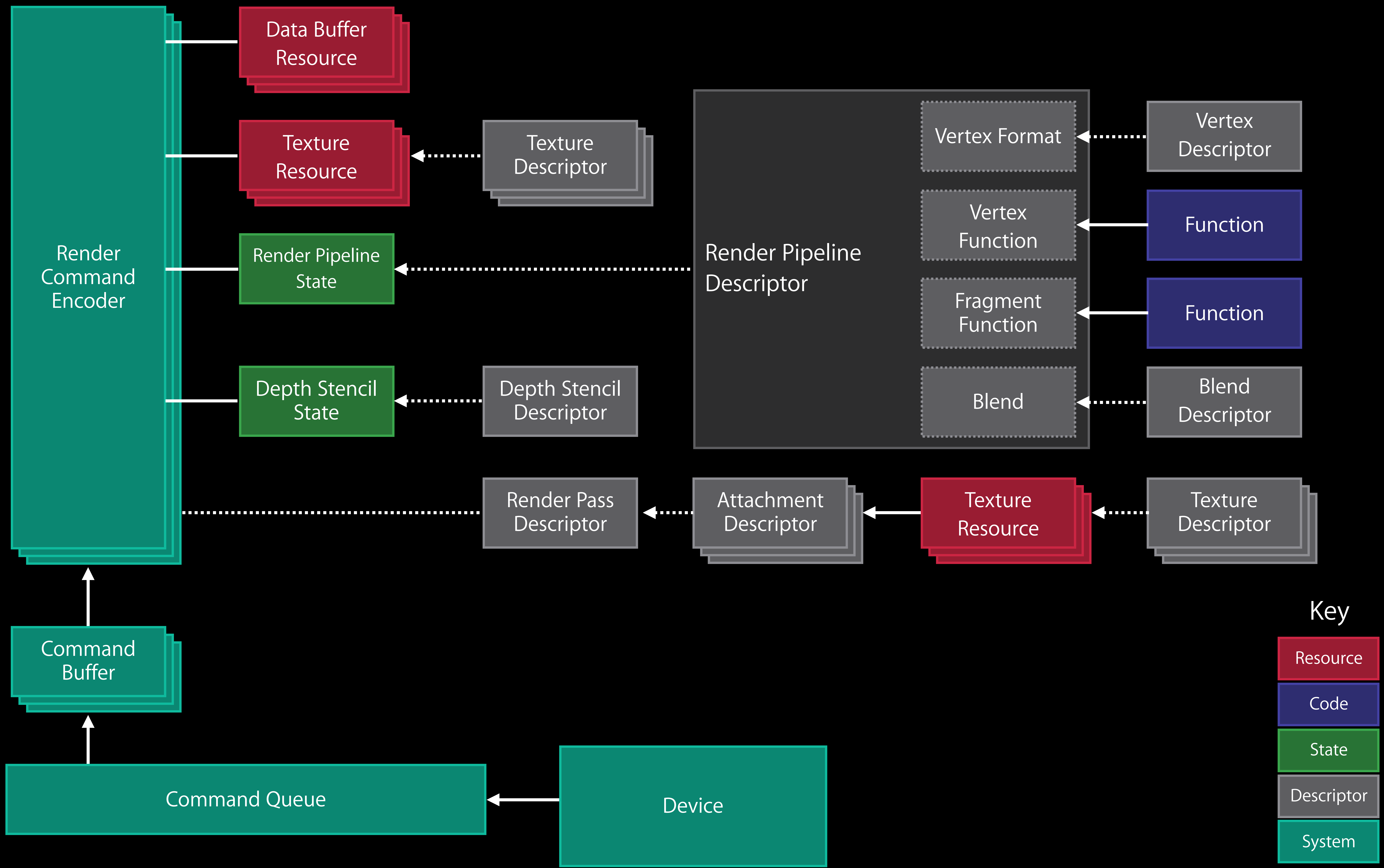

Data Buffer
Resource
Render
Command
Encoder
Command
Buffer
Command Queue
Device
Key
Resource
Code
State
Descriptor
System

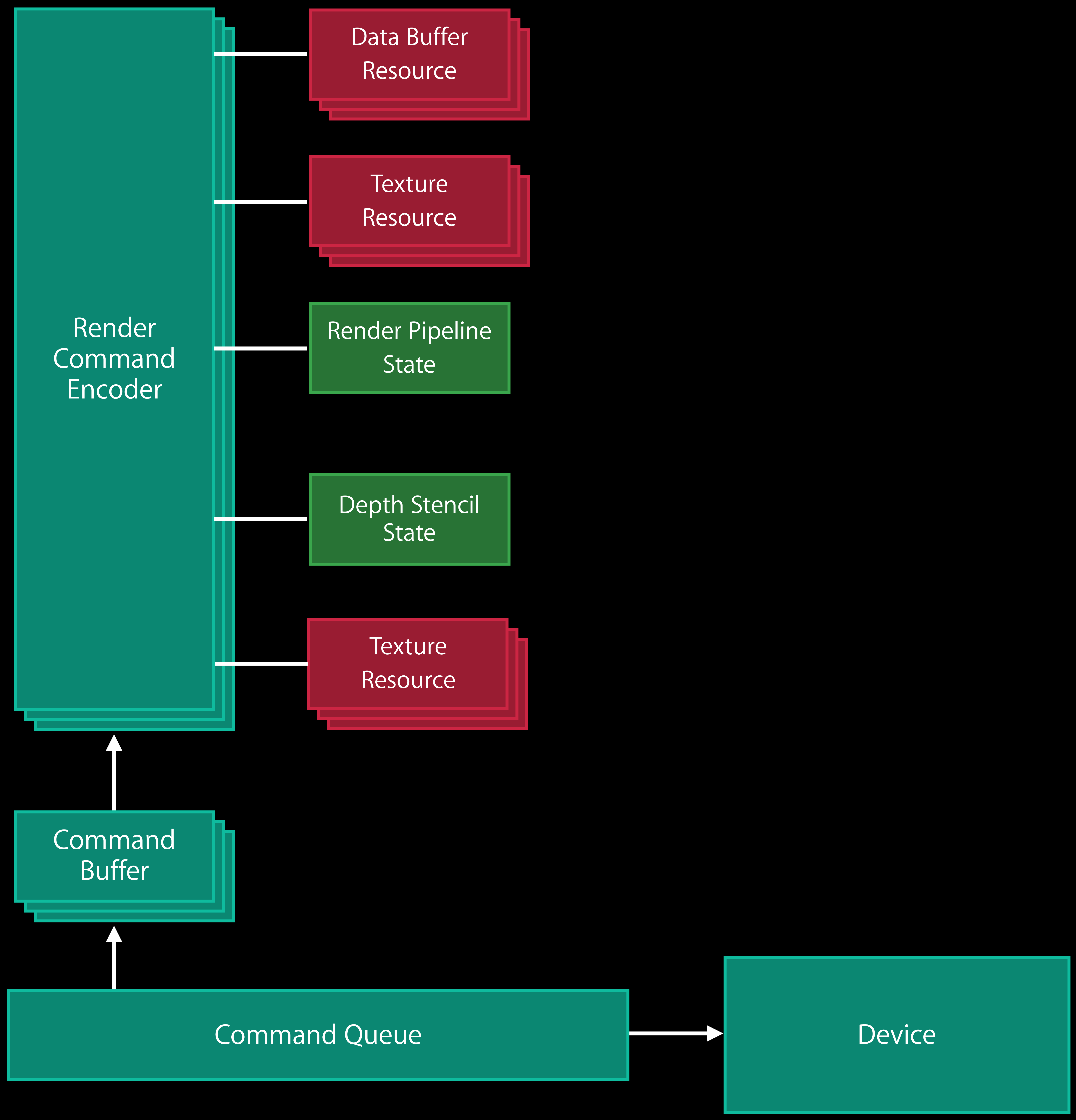

Data Buffer Resource
Texture Resource
Texture Descriptor
Render Command Encoder
Command Buffer
Command Queue
Device
Key
Resource
Code
State
Descriptor
System

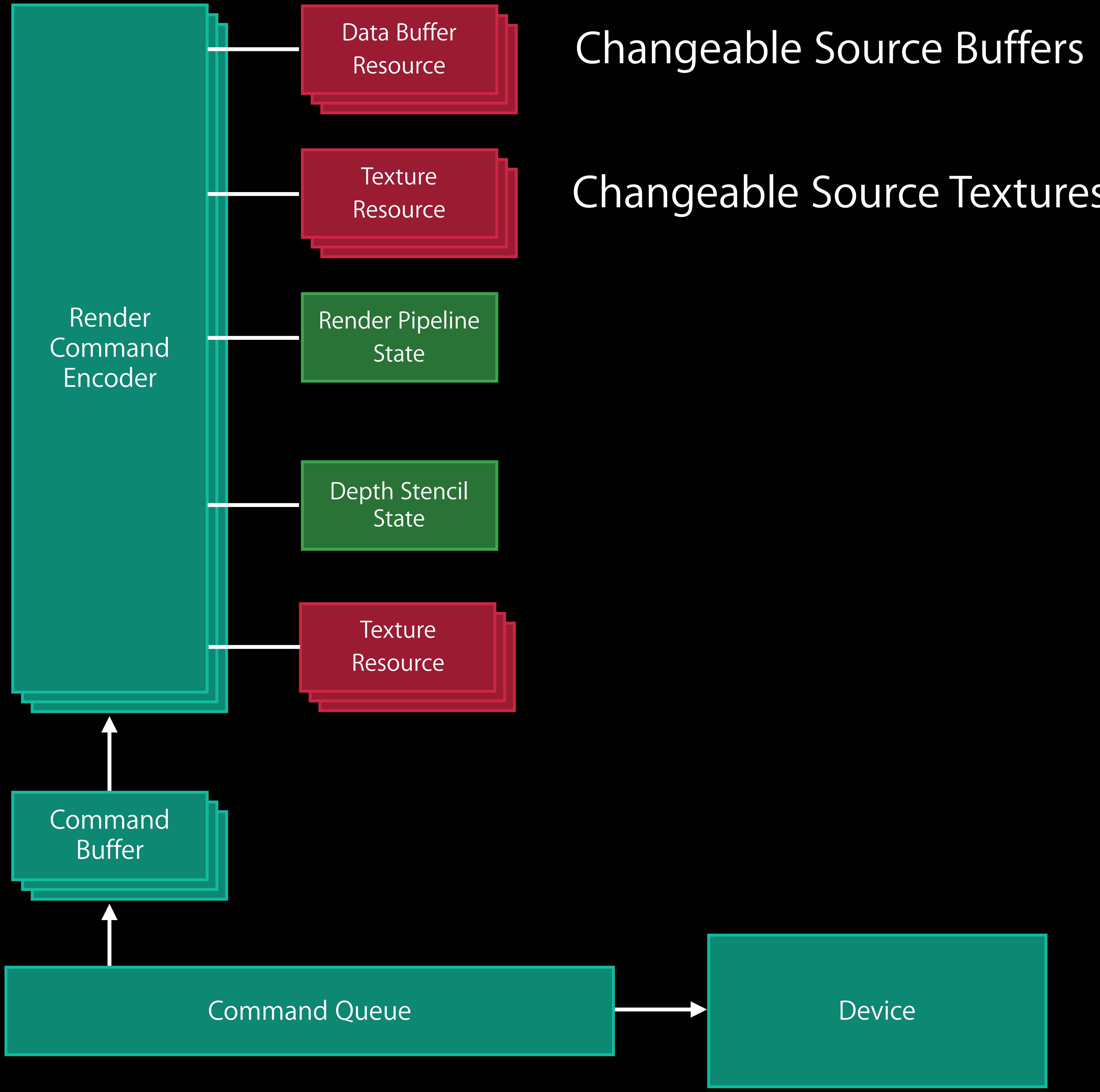


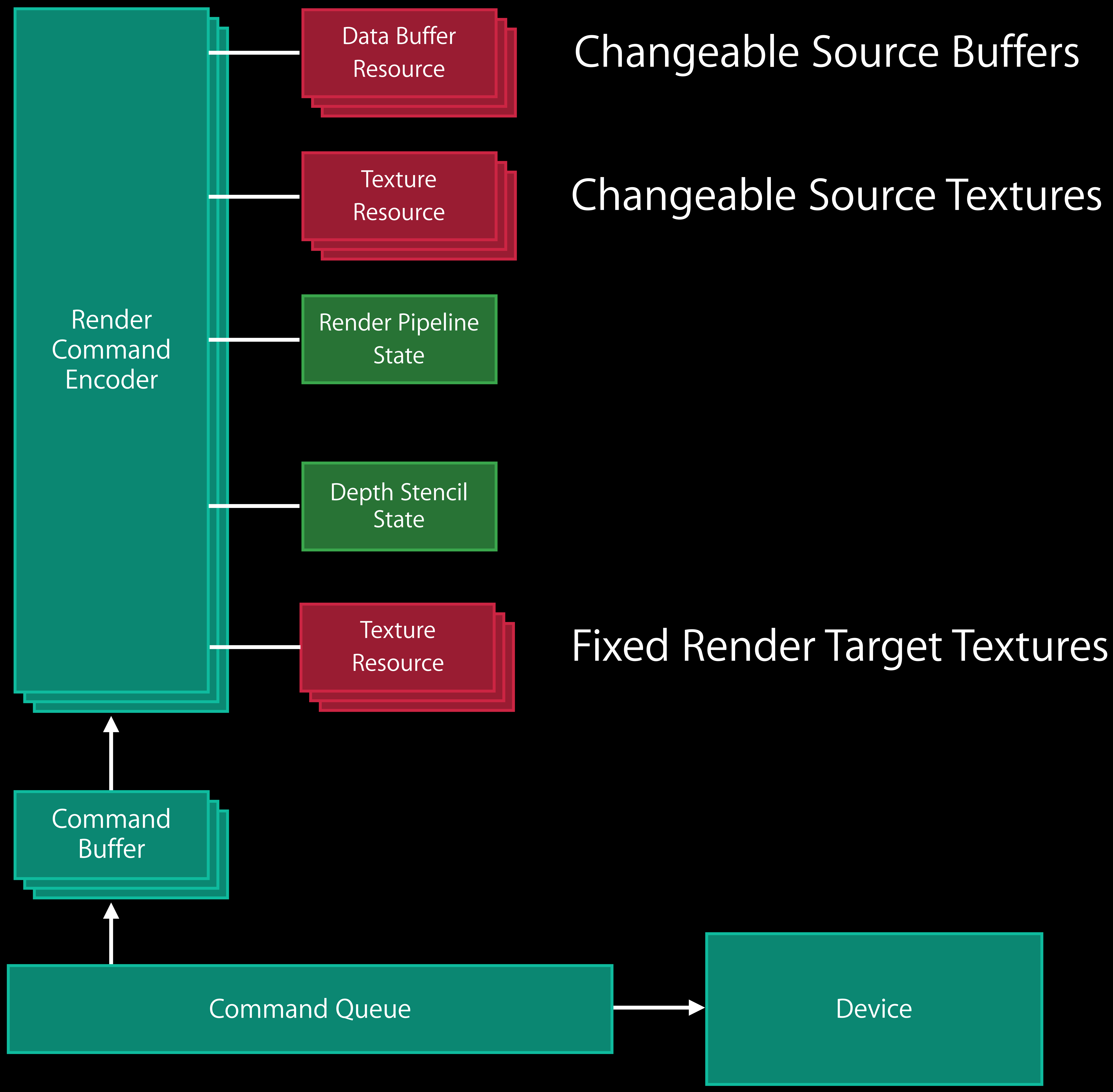
Data Buffer Resource
Texture Resource
Texture Descriptor
Render Pipeline State
Render Command Encoder
Render Pipeline Descriptor
Vertex Format
Vertex Descriptor
Vertex Function
Function
Fragment Function
Function
Blend
Blend Descriptor
Command Buffer
Command Queue
Device
Key
Resource
Code
State
Descriptor
System

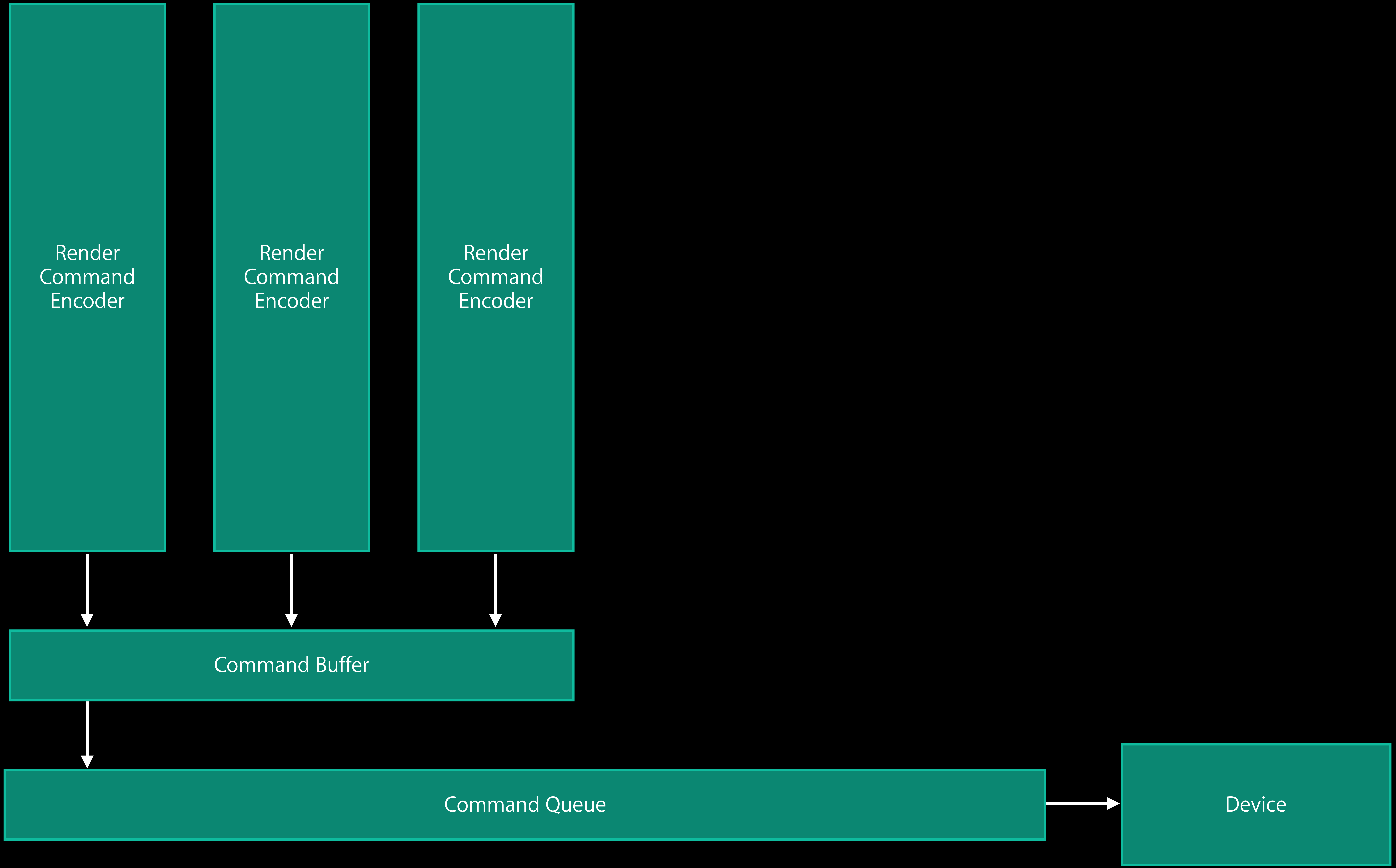

Data Buffer Resource
Texture Resource
Texture Descriptor
Render Pipeline State
Depth Stencil State
Depth Stencil Descriptor
Render Command Encoder
Render Pipeline Descriptor
Vertex Format
Vertex Descriptor
Vertex Function
Function
Fragment Function
Function
Blend
Blend Descriptor
Command Buffer
Command Queue
Device
Key
Resource
Code
State
Descriptor
System

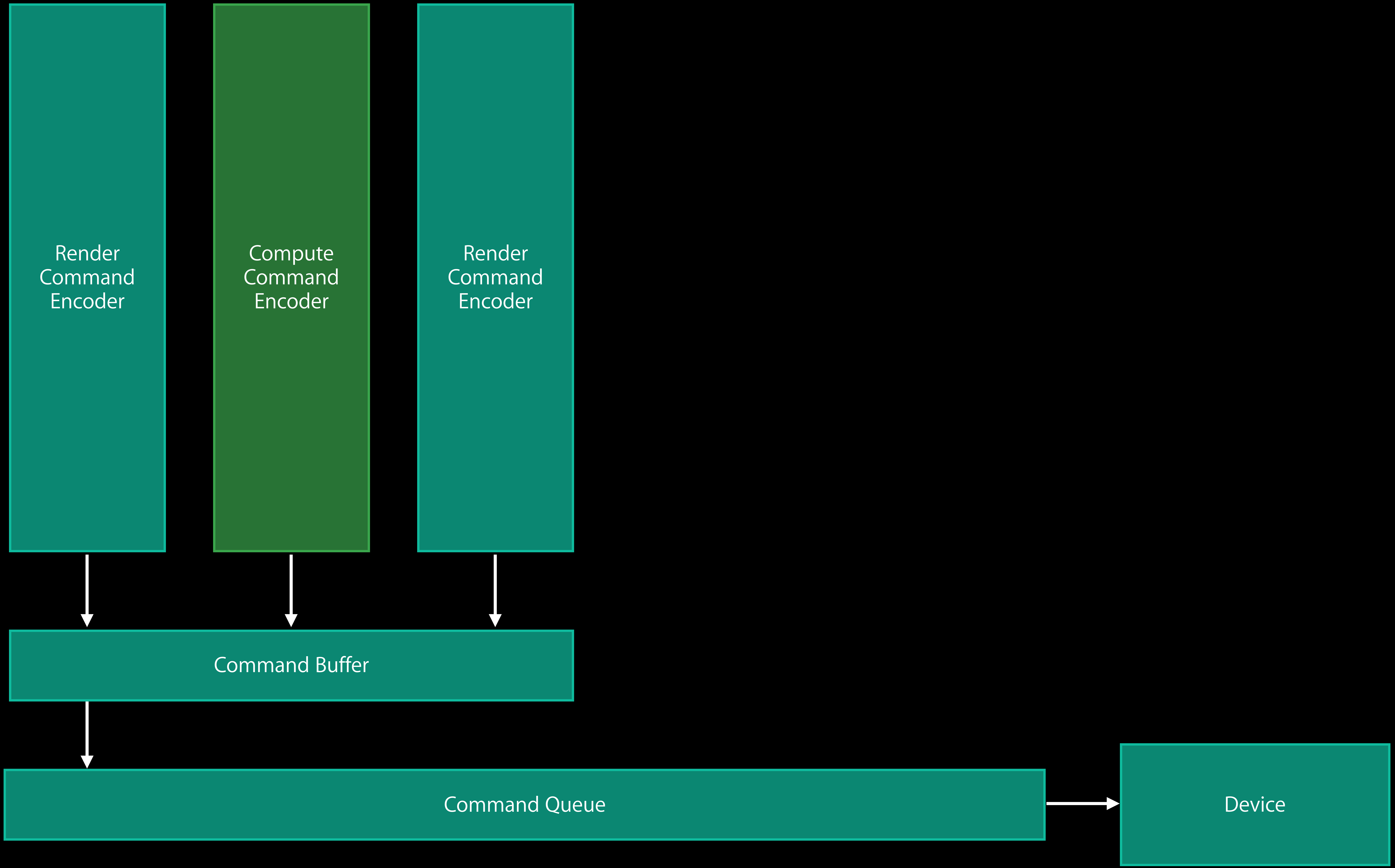

Data Buffer Resource
Texture Resource
Texture Descriptor
Render Pipeline State
Depth Stencil State
Depth Stencil Descriptor
Render Command Encoder
Render Pipeline Descriptor
Vertex Format
Vertex Descriptor
Vertex Function
Function
Fragment Function
Function
Blend
Blend Descriptor
Render Pass Descriptor
Attachment Descriptor
Texture Resource
Texture Descriptor
Command Buffer
Command Queue
Device
Key
Resource
Code
State
Descriptor
System

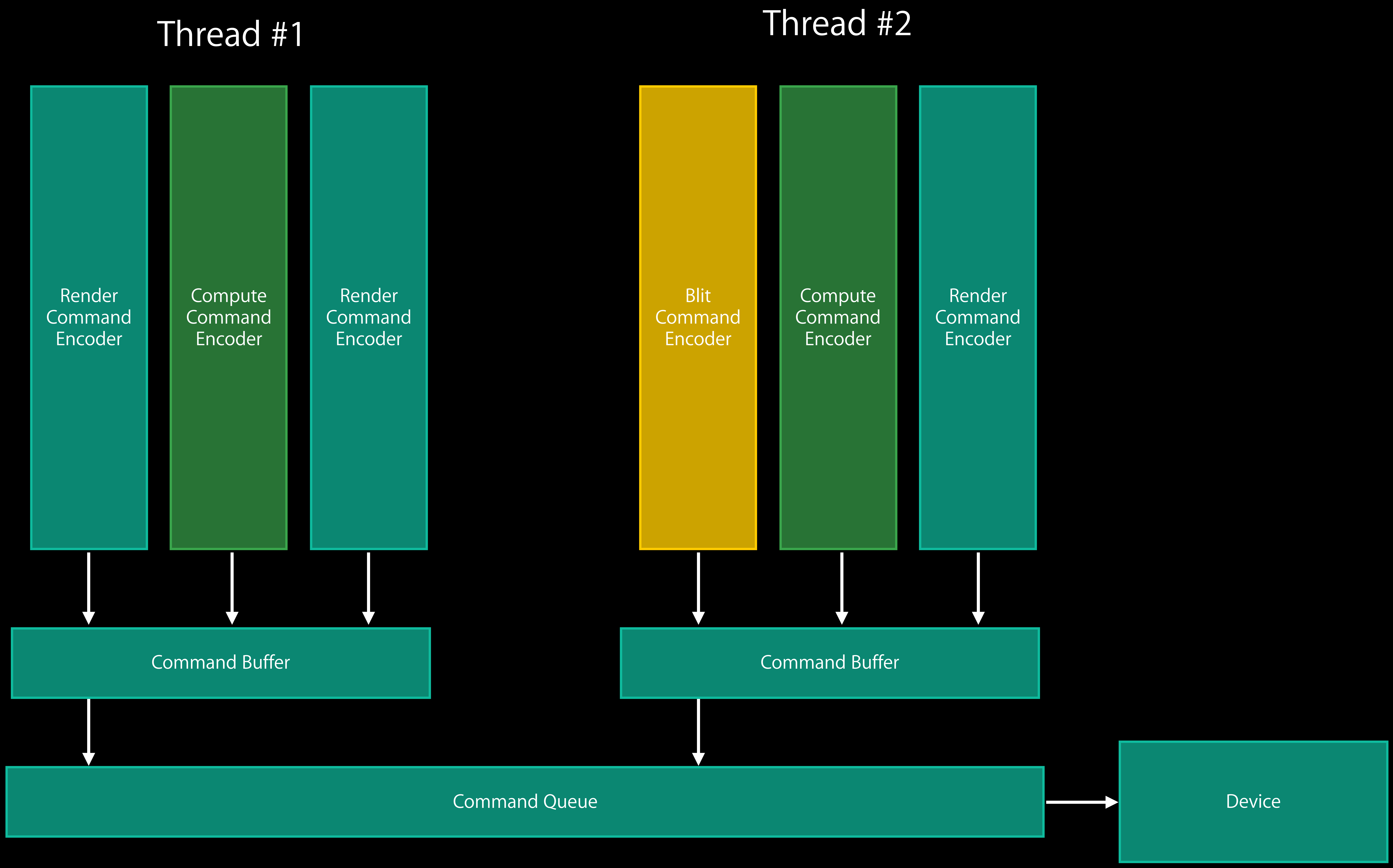

Data Buffer Resource
Texture Resource
Render Pipeline State
Depth Stencil State
Texture Resource
Render Command Encoder
Command Buffer
Command Queue
Device


Changeable Source Buffers
Changeable Source Textures
Render Command Encoder
Data Buffer Resource
Texture Resource
Render Pipeline State
Depth Stencil State
Texture Resource
Command Buffer
Command Queue
Device


Data Buffer Resource
Changeable Source Buffers
Texture Resource
Changeable Source Textures
Render Command Encoder
Render Pipeline State
Depth Stencil State
Texture Resource
Fixed Render Target Textures
Command Buffer
Command Queue
Device


Render Command Encoder
Render Command Encoder
Render Command Encoder
Command Buffer
Command Queue
Device


Render Command Encoder
Compute Command Encoder
Render Command Encoder
Command Buffer
Command Queue
Device


Thread #1
Thread #2
Render Command Encoder
Compute Command Encoder
Render Command Encoder
Blit Command Encoder
Compute Command Encoder
Render Command Encoder
Command Buffer
Command Buffer
Command Queue
Device

Command Submission Model

Command encoders convert API commands into hardware commands

Command Submission Model

Command encoders convert API commands into hardware commands

Hardware commands stored in command buffers

Command Submission Model

Command encoders convert API commands into hardware commands

Hardware commands stored in command buffers

Three types of command encoders

- Render, Compute, Blit
- Can interleave different types into single command buffer
- Avoids implicit expensive state save and restore operations

Command Submission Model

Command Submission Model

Explicit command buffer construction and submission

- App creates many lightweight command buffers
- App controls command buffer submission
- Metal signals app when command buffers finish execution

Command Submission Model

Explicit command buffer construction and submission

- App creates many lightweight command buffers
- App controls command buffer submission
- Metal signals app when command buffers finish execution

Command encoders generate commands immediately

- No deferred state validation
- Direct call to driver

Command Submission Model

Multithreaded command encoding

- Multiple command buffers can be encoded in parallel
- App decides execution order
- Very efficient implementation to ensure scalable performance

Resource Update Model

Resource Update Model

Designed for A7's unified memory system

- CPU and GPU share same storage
- No implicit copies
- Automatic CPU/GPU coherency model
  - CPU and GPU observe writes at command buffer execution boundaries
  - No explicit CPU cache management required

Resource Update Model

Designed for A7's unified memory system

- CPU and GPU share same storage
- No implicit copies
- Automatic CPU/GPU coherency model
  - CPU and GPU observe writes at command buffer execution boundaries
  - No explicit CPU cache management required

Significantly higher performance

- More synchronization responsibilities for you

Resource Update Model

Resource Update Model

Two types of resources

- Textures (formatted images)
  - Render targets, texture sources
- Data buffers (unformatted memory)
  - "a bag of bytes"
  - Vertex data, shader constants, output memory, etc.

Resource Update Model

Two types of resources

- Textures (formatted images)
  - Render targets, texture sources
- Data buffers (unformatted memory)
  - "a bag of bytes"
  - Vertex data, shader constants, output memory, etc.

Resource structure (size, levels, format) can't be changed

- Avoids costly resource validation
- Resource contents can be changed

Resource Update Model

Resource Update Model

Updating data buffers

- Direct access to storage by CPU
- No “lock” API needed

Resource Update Model

Updating data buffers

- Direct access to storage by CPU
- No “lock” API needed

Updating textures

- Implementation private storage
- Metal provides blazing fast texture update routines

Resource Update Model

Updating data buffers

- Direct access to storage by CPU
- No “lock” API needed

Updating textures

- Implementation private storage
- Metal provides blazing fast texture update routines

GPU-accelerated and pipelined updates

- Via Blit command encoder

Resource Update Model

Resource Update Model

Can share texture storage with other textures

- Interpret as different pixel formats with same pixel size
  - eg., sRGB vs. RGB, or single 32-bit component vs. RGBA8888

Resource Update Model

Can share texture storage with other textures

- Interpret as different pixel formats with same pixel size
  - eg., sRGB vs. RGB, or single 32-bit component vs. RGBA8888

Can share texture storage with other buffers

- Assumes "row-linear" pixel data

Command Encoder Types

Command Encoder Types

Render command encoder

- Graphics rendering

Command Encoder Types

Render command encoder

- Graphics rendering

Compute command encoder

- Data parallel computations

Command Encoder Types

Render command encoder

- Graphics rendering

Compute command encoder

- Data parallel computations

Blit command encoder

- GPU-accelerated resource copy operations

Render Command Encoder

Render Command Encoder

Generates hardware commands for single rendering “pass”

- All rendering to single framebuffer object
- Specifies states for vertex and fragment stages of 3D pipeline
- Interleaves resources, state changes, and draw calls

Render Command Encoder

Generates hardware commands for single rendering “pass”

- All rendering to single framebuffer object
- Specifies states for vertex and fragment stages of 3D pipeline
- Interleaves resources, state changes, and draw calls

No “draw time” compilation

- App controls when all significant compilation and state validation occurs

Render State Objects

State Object	Description
DepthStencil	DepthStencil comparison functions and write masks
Sampler	Filter states, addressing modes, LOD state
Render Pipeline	"Everything else" Vertex and pixel shader functions Vertex data layout Multisample state Blend state Color write masks

Render States

States affecting compilation can't be changed after object creation

Inexpensive states can be changed

Changeable	Render States
✗	Vertex and fragment shaders # of render targets, pixel format, color write masks Multisample state Blend state DepthStencil state and write masks
✓	Specification of buffers, textures, samplers Cull mode and facing orientation Depth clipping and depth bias Polygon mode Viewport and scissor Occlusion queries

Framebuffer Loads and Stores

Framebuffer configuration designed for optimal behavior on A7 GPU

- Tile-based deferred-mode renderer

Framebuffer Loads and Stores

Framebuffer configuration designed for optimal behavior on A7 GPU

- Tile-based deferred-mode renderer

Explicit control of framebuffer tile cache "Load and Store" operations

- Load at start of render pass
- Store at end of render pass

Framebuffer Loads and Stores

Framebuffer configuration designed for optimal behavior on A7 GPU

- Tile-based deferred-mode renderer

Explicit control of framebuffer tile cache “Load and Store” operations

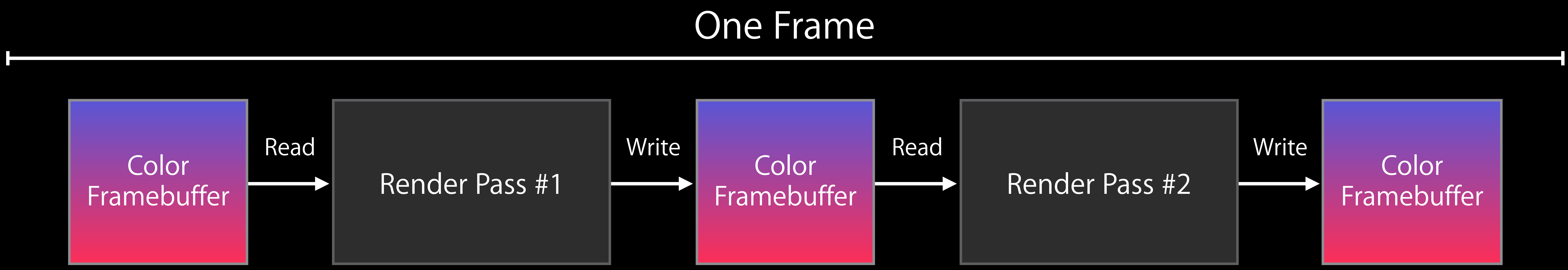
- Load at start of render pass
- Store at end of render pass

App choses per render target

- Load—Don’t care, load, clear
- Store—Don’t care, store, multisample resolve

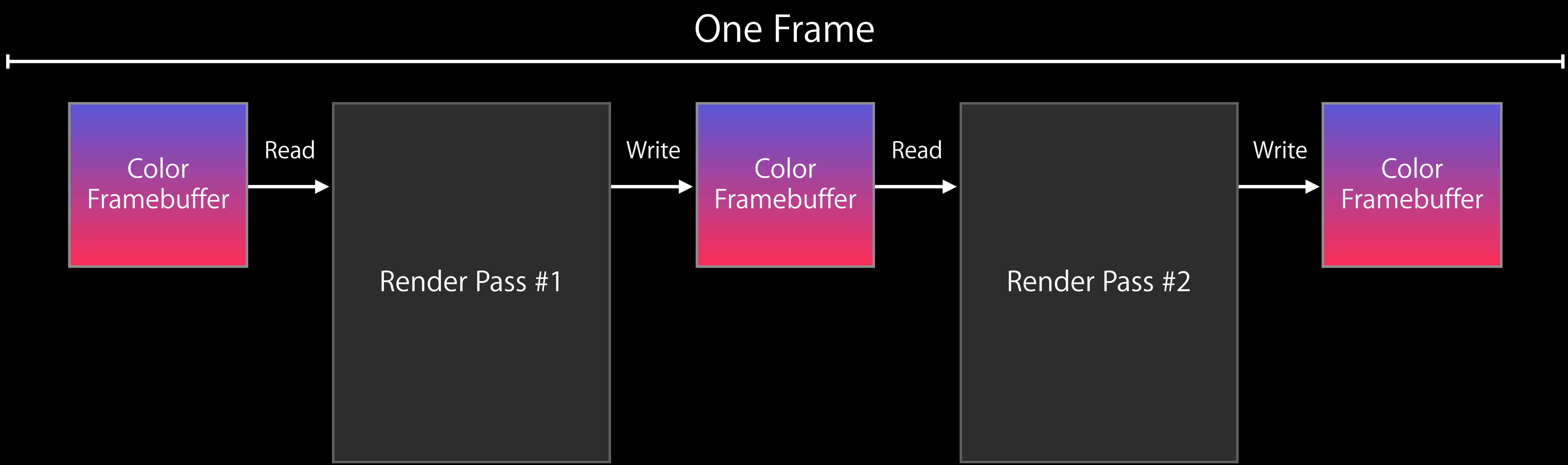
Before Metal

Framebuffer loads and stores



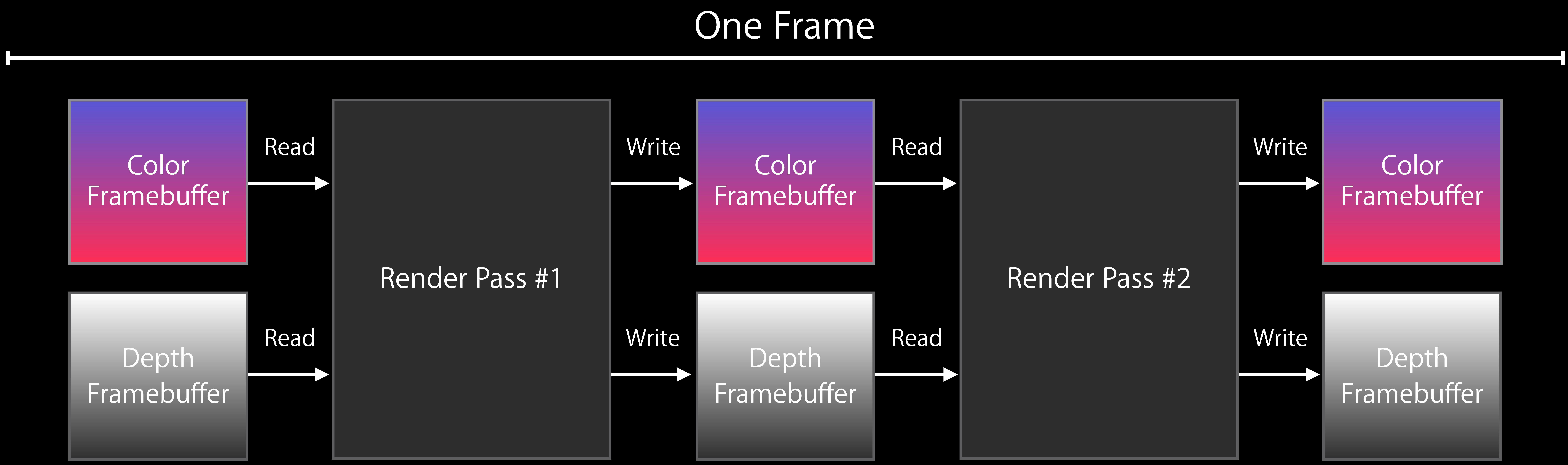
Before Metal

Framebuffer loads and stores



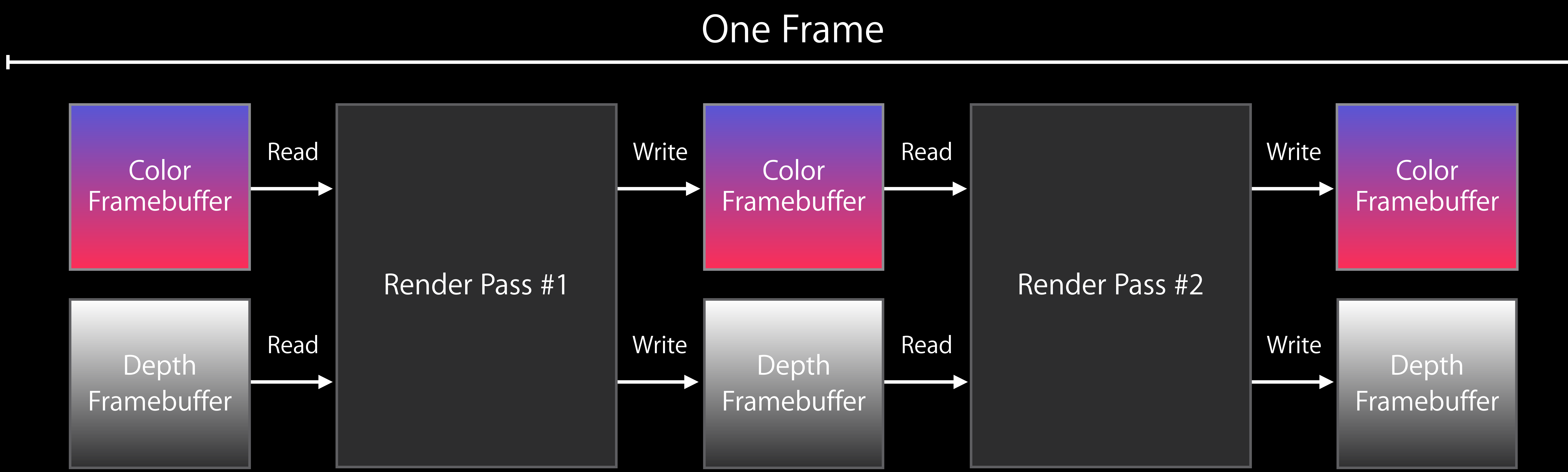
Before Metal

Framebuffer loads and stores



Before Metal

Framebuffer loads and stores

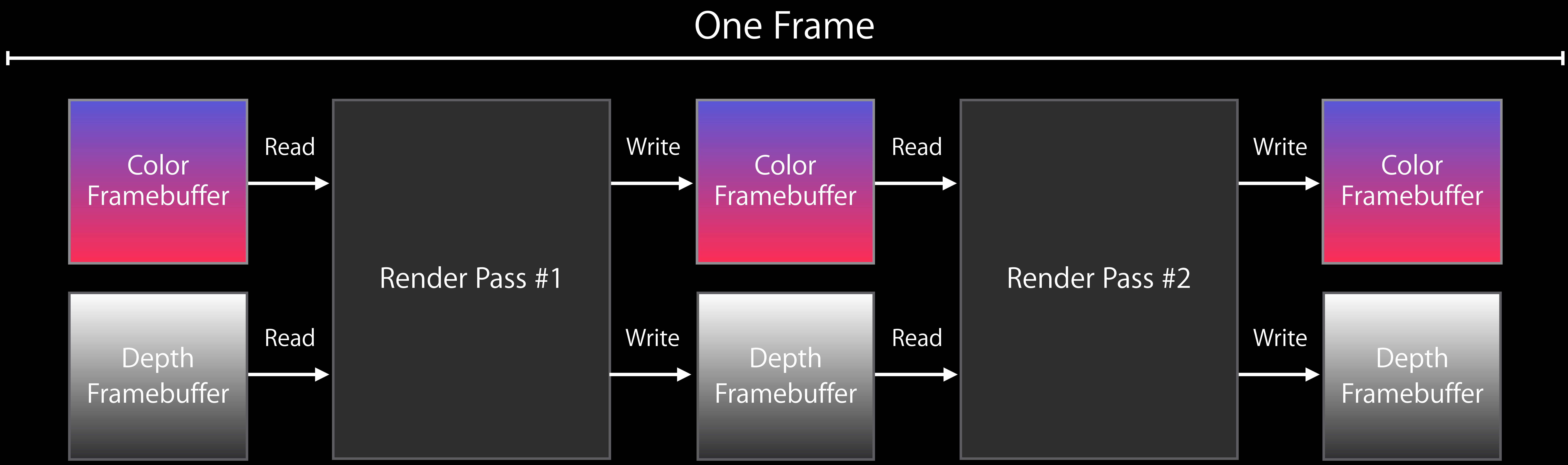


Framebuffer Memory Bandwidth

Color: 2 reads + 2 writes
Depth: 2 reads + 2 writes

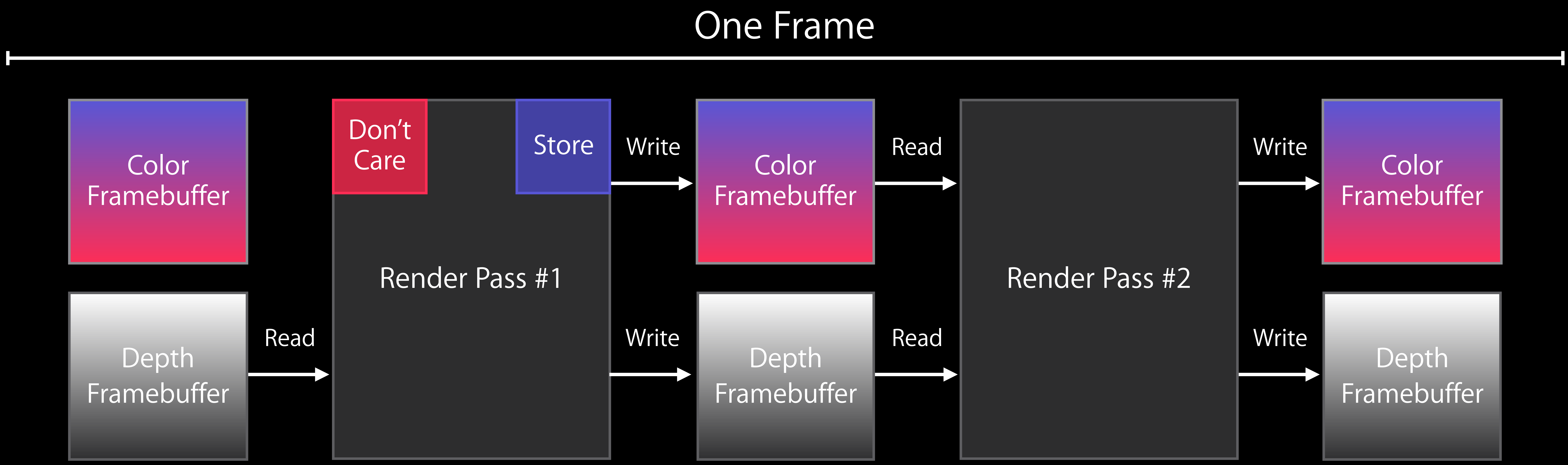
Metal

Framebuffer loads and stores



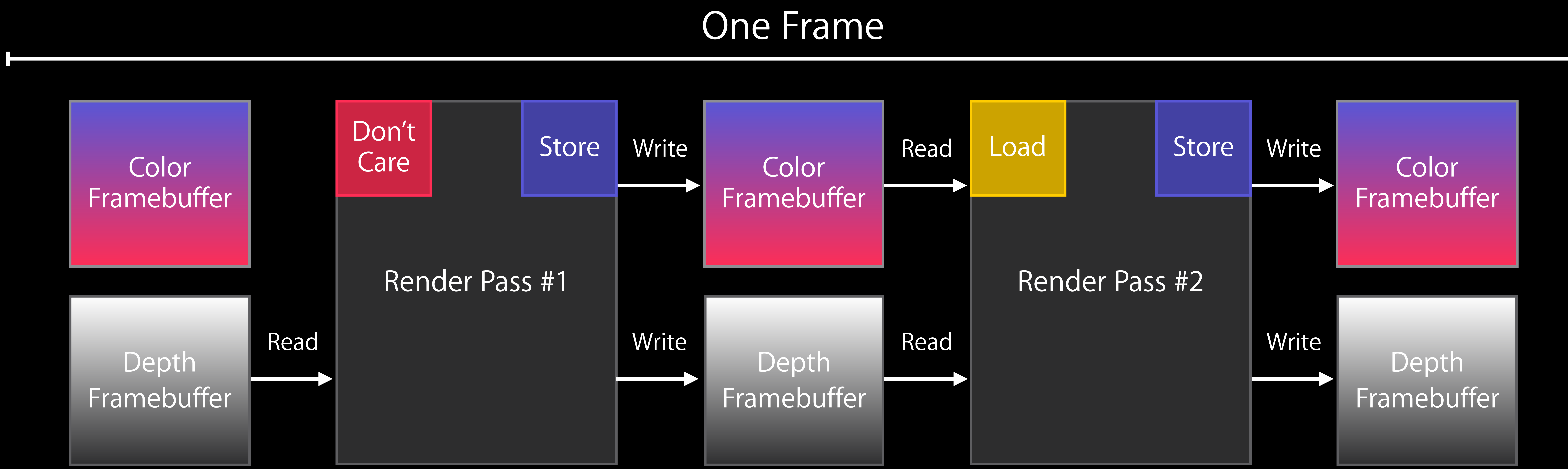
Metal

Framebuffer loads and stores



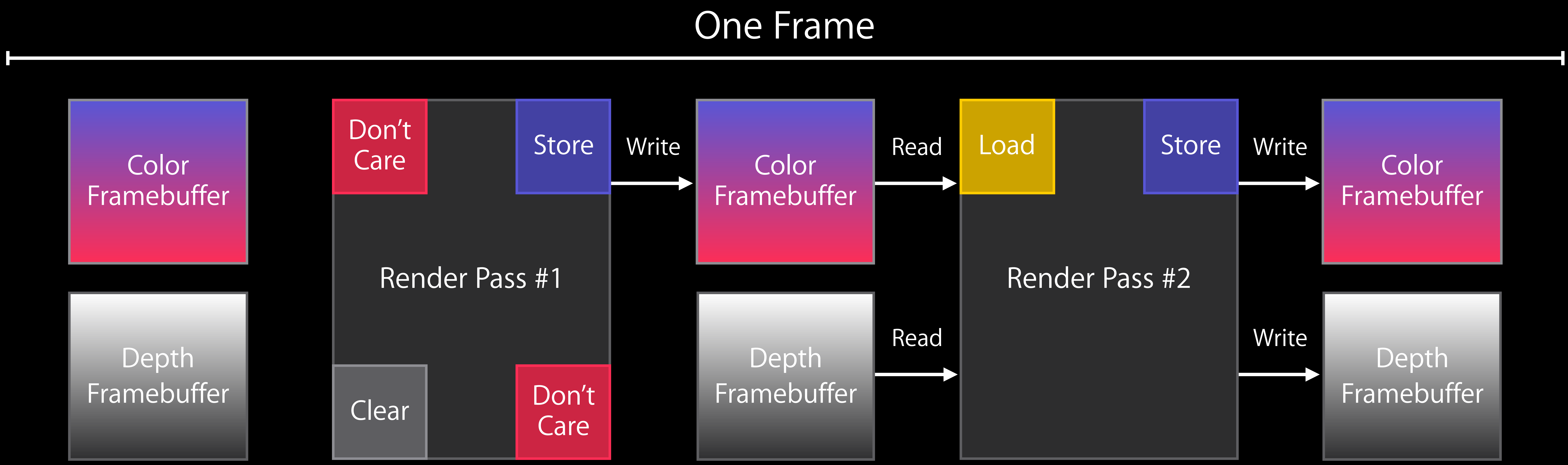
Metal

Framebuffer loads and stores



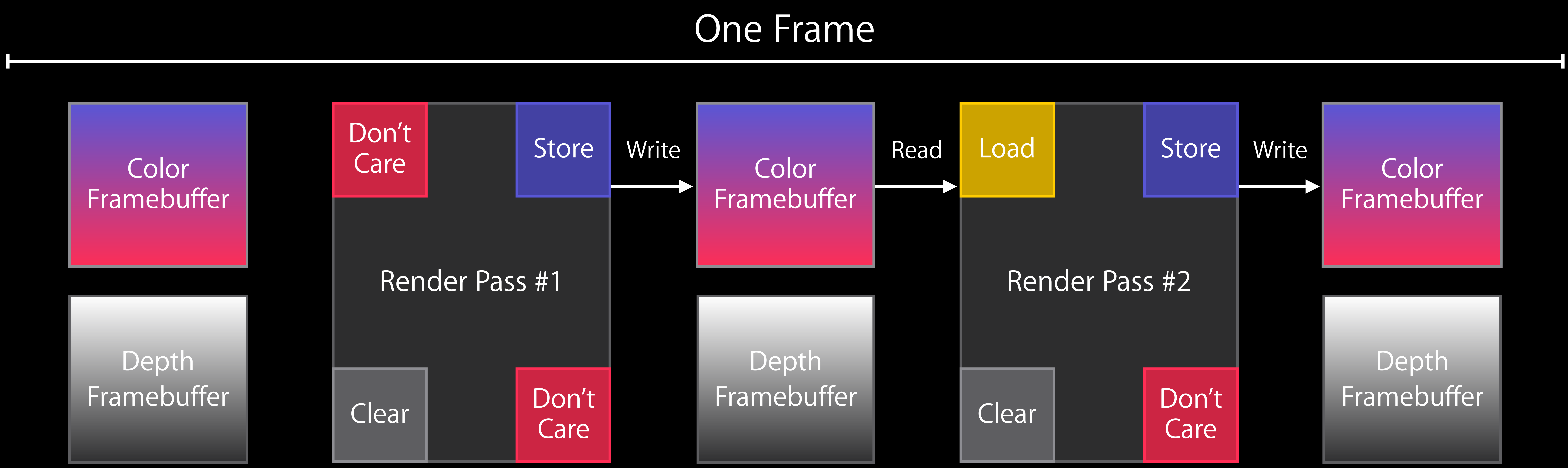
Metal

Framebuffer loads and stores

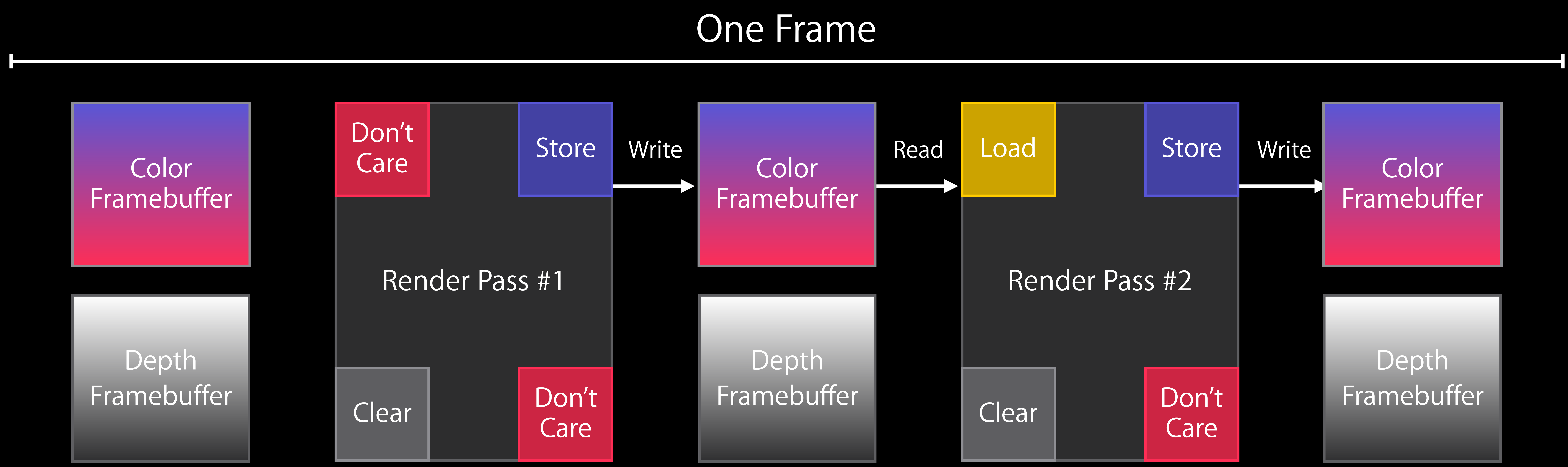


Metal

Framebuffer loads and stores



Dramatic Reduction in Memory Traffic



Framebuffer Memory Bandwidth

Color: 1 reads + 2 writes
Depth: 0 reads + 0 writes

Compute Command Encoder

Familiar compute run-time and memory model

- Textures and data buffers
- Local and global memory
- Local atomics
- Barriers
- Memory loads and stores
- User-specified workgroup dimensions

Compute Command Encoder

Compute Command Encoder

Fully integrated with graphics

- Unified API, shading language and developer tools
- Efficiently interleaves Compute commands with Render and Blit commands

Compute Command Encoder

Fully integrated with graphics

- Unified API, shading language and developer tools
- Efficiently interleaves Compute commands with Render and Blit commands

No “execution-time” compilation

- App controls when all significant compilation and validation occurs

Compute Command Encoder

State Object	Description
Compute State	Compute functions, workgroup configuration
Sampler	Filter states, addressing modes, LOD state

Blit Command Encoder

Enables asynchronous copies

- In parallel with graphics and compute operations

Blit Command Encoder

Enables asynchronous copies

- In parallel with graphics and compute operations

Texture uploads

- Copy to/from other texture or data buffer
- Accelerated mipmap generation

Blit Command Encoder

Enables asynchronous copies

- In parallel with graphics and compute operations

Texture uploads

- Copy to/from other texture or data buffer
- Accelerated mipmap generation

Data Buffer updates

- Copy to/from another data buffer or texture
- Fill with constant values

Agenda

Background

API concepts

Shading language

Developer tools

Shading Language

Shading Language

Unified shading language for graphics and compute processing

Shading Language

Unified shading language for graphics and compute processing

Based on C++11

- Static subset
- Built from LLVM and clang

Shading Language

Unified shading language for graphics and compute processing

Based on C++11

- Static subset
- Built from LLVM and clang

Additions

- GPU hardware features (texture sampling, rasterization, compute operations, etc.)
- Function overloading and templates

Shading Language

Shading Language

Data types for graphics and compute features

- Scalar, vector and matrix types
- Samplers and textures

Shading Language

Data types for graphics and compute features

- Scalar, vector and matrix types
- Samplers and textures

“Attributes”

- Function arguments
- Sampling and interpolation qualifiers
- Per-instance inputs, outputs, and built-in graphics variables
- Programmable blending

Shading Language

Shading Language

Multiple shaders per source file

Shading Language

Multiple shaders per source file

Metal shaders built by Xcode compiler into Metal library files

- Library contains archive of Metal shaders
- With run-time APIs
  - Load a Metal library
  - Finalize compilation to GPU machine code

Shading Language

Multiple shaders per source file

Metal shaders built by Xcode compiler into Metal library files

- Library contains archive of Metal shaders
- With run-time APIs
  - Load a Metal library
  - Finalize compilation to GPU machine code

Metal includes standard library for graphics and compute functions

Argument Tables

Textures, buffers, and samplers passed as arguments to functions

- Vertex, fragment, compute shaders

Argument Tables

Textures, buffers, and samplers passed as arguments to functions

- Vertex, fragment, compute shaders

Each command encoder includes set of "argument tables"

- One table per type (texture, buffer, sampler)

Argument Tables

Textures, buffers, and samplers passed as arguments to functions

- Vertex, fragment, compute shaders

Each command encoder includes set of "argument tables"

- One table per type (texture, buffer, sampler)

Metal API and shading language use table index to reference arguments

Argument Tables

Texture Index	ID
0	Texture A
1	Texture B
..	Texture ..
n	Texture Z

Command Encoder

Argument Tables

Texture Index	ID
0	Texture A
1	Texture B
..	Texture ..
n	Texture Z

Buffer Index	ID
0	Buffer A
1	Buffer B
..	Buffer ..
n	Buffer Z

Sampler Index	ID
0	Sampler A
1	Sampler B
..	Sampler ..
n	Sampler Z

Command Encoder

Argument Tables

```
vertex VertexOutput
smoothTriangleVertex(constant float4 *pos_data [[ buffer(0) ]],
                     constant float2 *uv_data  [[ buffer(1) ]],
                     uint vid [[ vertex_id ]])
{
    VertexOutput out;
    out.pos = pos_data[vid];
    out.uv = uv_data[vid];
    return out;
}

fragment float4
smoothTriangleFragment(VertexOutput in [[ stage_in ]],
                       texture2d<float> tex [[ texture(1) ]])
{
    return tex.sample(s, in.uv);
}
```

Shader Code

Texture Index	ID
0	Texture A
1	Texture B
..	Texture ..
n	Texture Z

Buffer Index	ID
0	Buffer A
1	Buffer B
..	Buffer ..
n	Buffer Z

Sampler Index	ID
0	Sampler A
1	Sampler B
..	Sampler ..
n	Sampler Z

Command Encoder

Argument Tables

```
vertex VertexOutput
smoothTriangleVertex(constant float4 *pos_data [[ buffer(0) ]],
                     constant float2 *uv_data  [[ buffer(1) ]],
                     uint vid [[ vertex_id ]])
{
    VertexOutput out;
    out.pos = pos_data[vid];
    out.uv = uv_data[vid];
    return out;
}

fragment float4
smoothTriangleFragment(VertexOutput in [[ stage_in ]],
                       texture2d<float> tex [[ texture(1) ]])
{
    return tex.sample(s, in.uv);
}
```

Shader Code

Texture Index	ID
0	Texture A
1	Texture B
..	Texture ..
n	Texture Z

Buffer Index	ID
0	Buffer A
1	Buffer B
..	Buffer ..
n	Buffer Z

Sampler Index	ID
0	Sampler A
1	Sampler B
..	Sampler ..
n	Sampler Z

Command Encoder

Argument Tables

```
vertex VertexOutput
smoothTriangleVertex(constant float4 *pos_data [[ buffer(0) ]],
                     constant float2 *uv_data  [[ buffer(1) ]],
                     uint vid [[ vertex_id ]])
{
    VertexOutput out;
    out.pos = pos_data[vid];
    out.uv = uv_data[vid];
    return out;
}

fragment float4
smoothTriangleFragment(VertexOutput in [[ stage_in ]],
                       texture2d<float> tex [[ texture(1) ]])
{
    return tex.sample(s, in.uv);
}
```

Shader Code

Texture Index	ID
0	Texture A
1	Texture B
..	Texture ..
n	Texture Z

Buffer Index	ID
0	Buffer A
1	Buffer B
..	Buffer ..
n	Buffer Z

Sampler Index	ID
0	Sampler A
1	Sampler B
..	Sampler ..
n	Sampler Z

Command Encoder

Argument Tables

```
vertex VertexOutput
smoothTriangleVertex(constant float4 *pos_data [[ buffer(0) ]],
                     constant float2 *uv_data  [[ buffer(1) ]],
                     uint vid [[ vertex_id ]])
{
    VertexOutput out;
    out.pos = pos_data[vid];
    out.uv = uv_data[vid];
    return out;
}

fragment float4
smoothTriangleFragment(VertexOutput in [[ stage_in ]],
                       texture2d<float> tex [[ texture(1) ]])
{
    return tex.sample(s, in.uv);
}
```

Shader Code

Texture Index	ID
0	Texture A
1	Texture B
..	Texture ..
n	Texture Z

Buffer Index	ID
0	Buffer A
1	Buffer B
..	Buffer ..
n	Buffer Z

Sampler Index	ID
0	Sampler A
1	Sampler B
..	Sampler ..
n	Sampler Z

Command Encoder

Agenda

- Background
- API concepts
- Shading language
- Developer tools

Metal Shader Compiler Process

Metal Shader Compiler Process

Metal shader sources compiled to libraries at application build time

- No need to ship source code with application
- Shading language errors reported at build time

Metal Shader Compiler Process

Metal shader sources compiled to libraries at application build time

- No need to ship source code with application
- Shading language errors reported at build time

Metal libraries compiled to device code at state object creation time

- No draw time compilation
- Device code cached after compilation

Metal Shader Compiler Process

Metal shader sources compiled to libraries at application build time

- No need to ship source code with application
- Shading language errors reported at build time

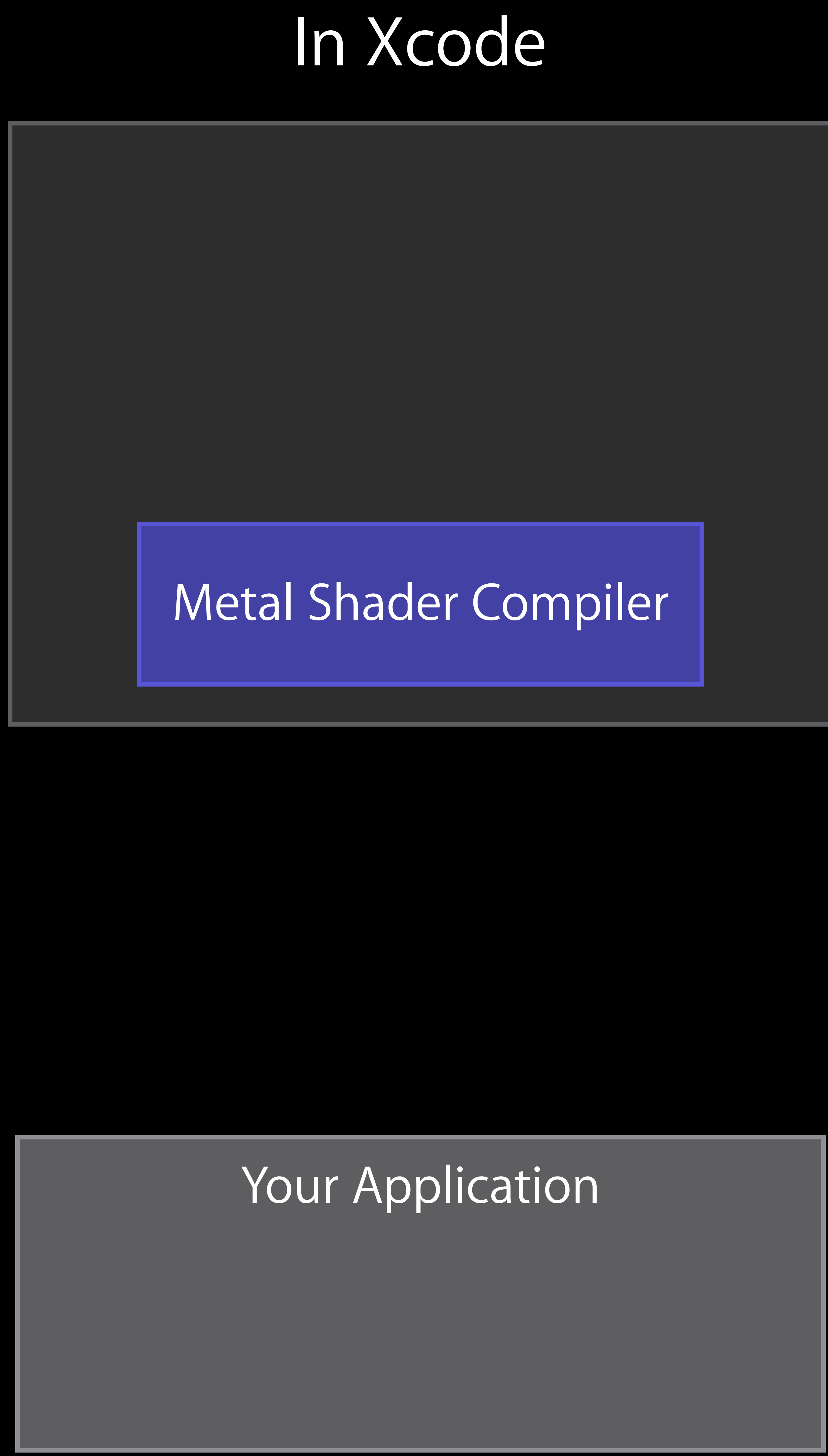
Metal libraries compiled to device code at state object creation time

- No draw time compilation
- Device code cached after compilation

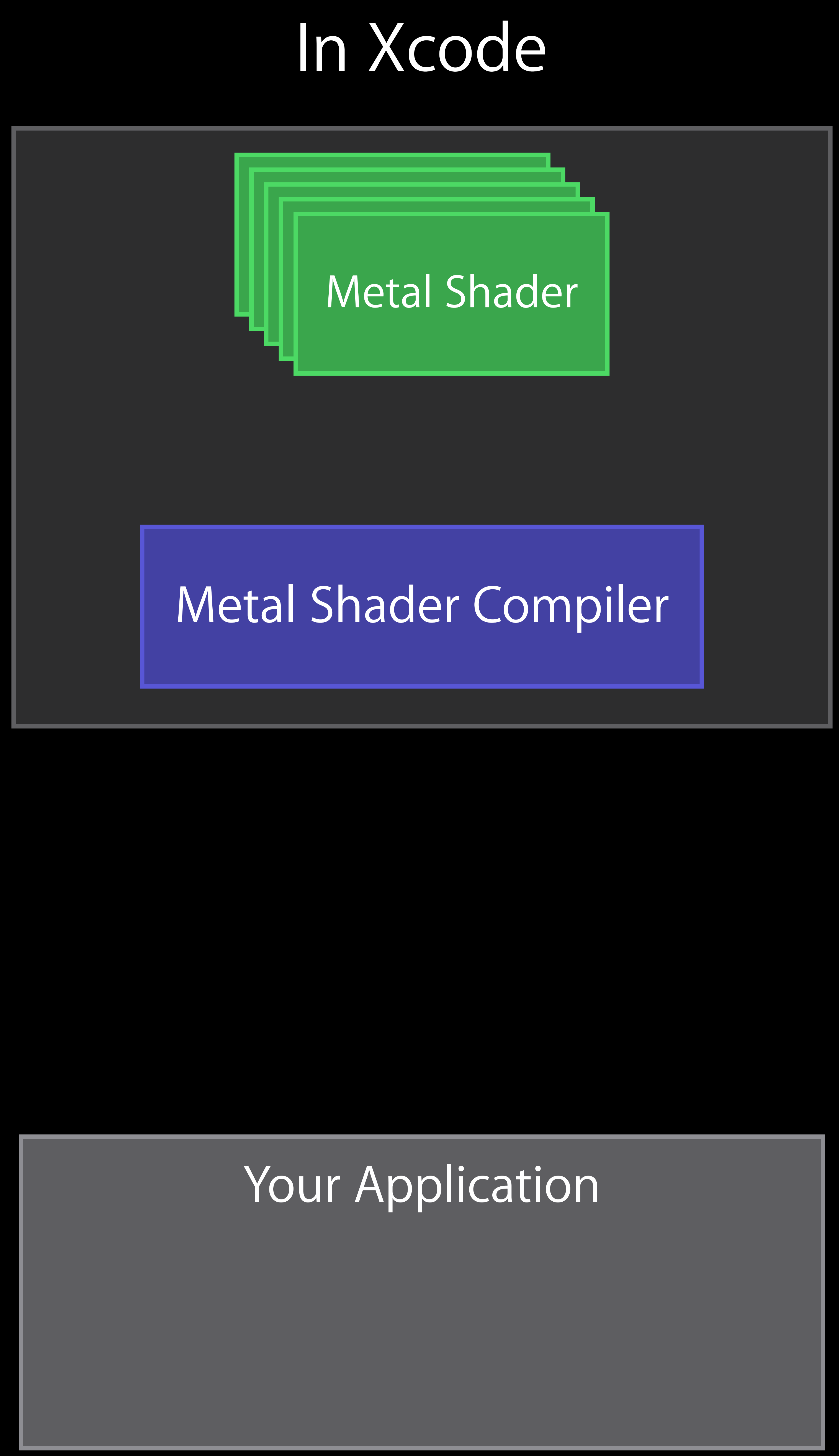
There is also a run-time shader compiler

- No draw time compilation
- For best performance, use offline compiler

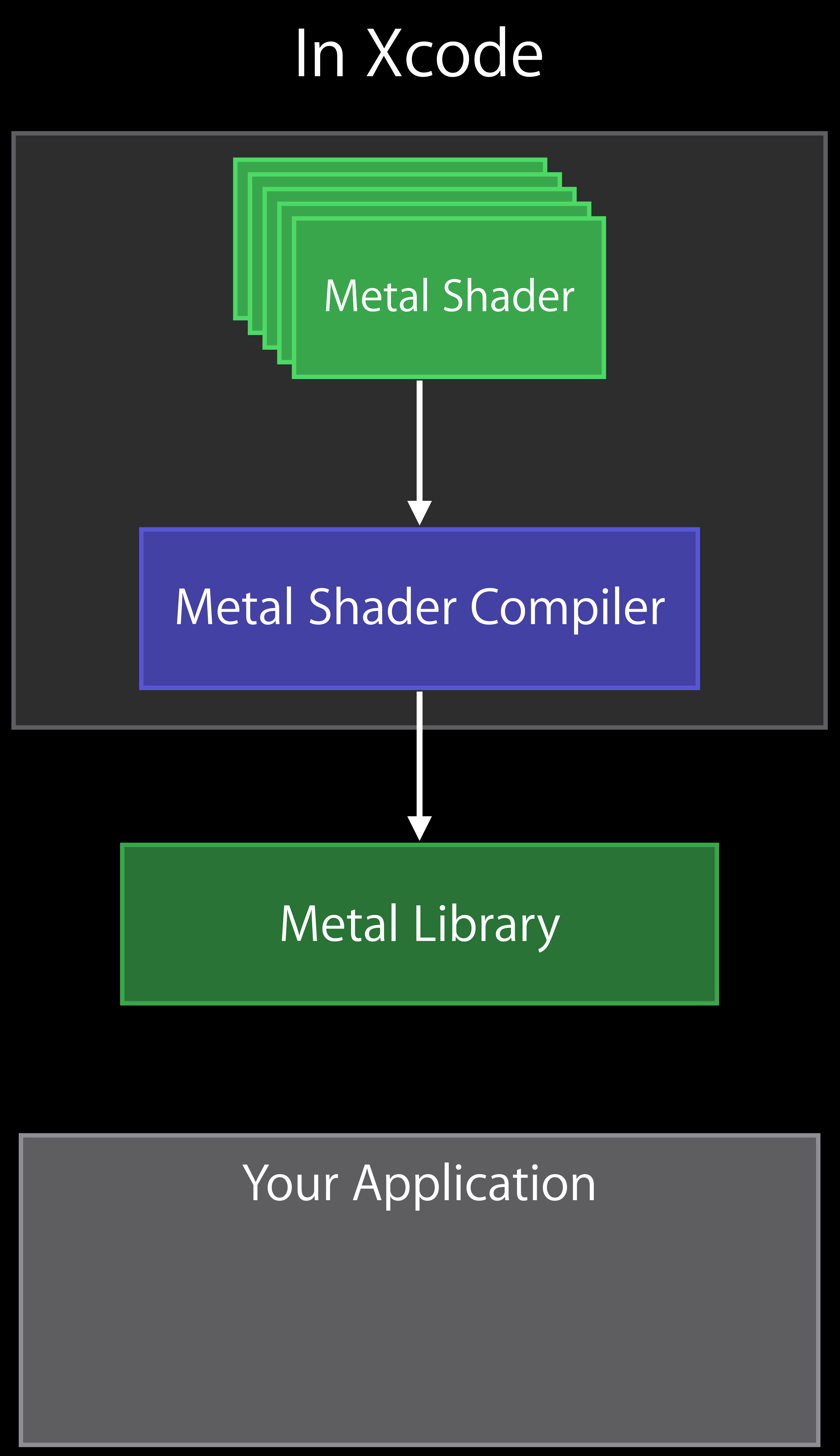
Metal Shader Compiler Process



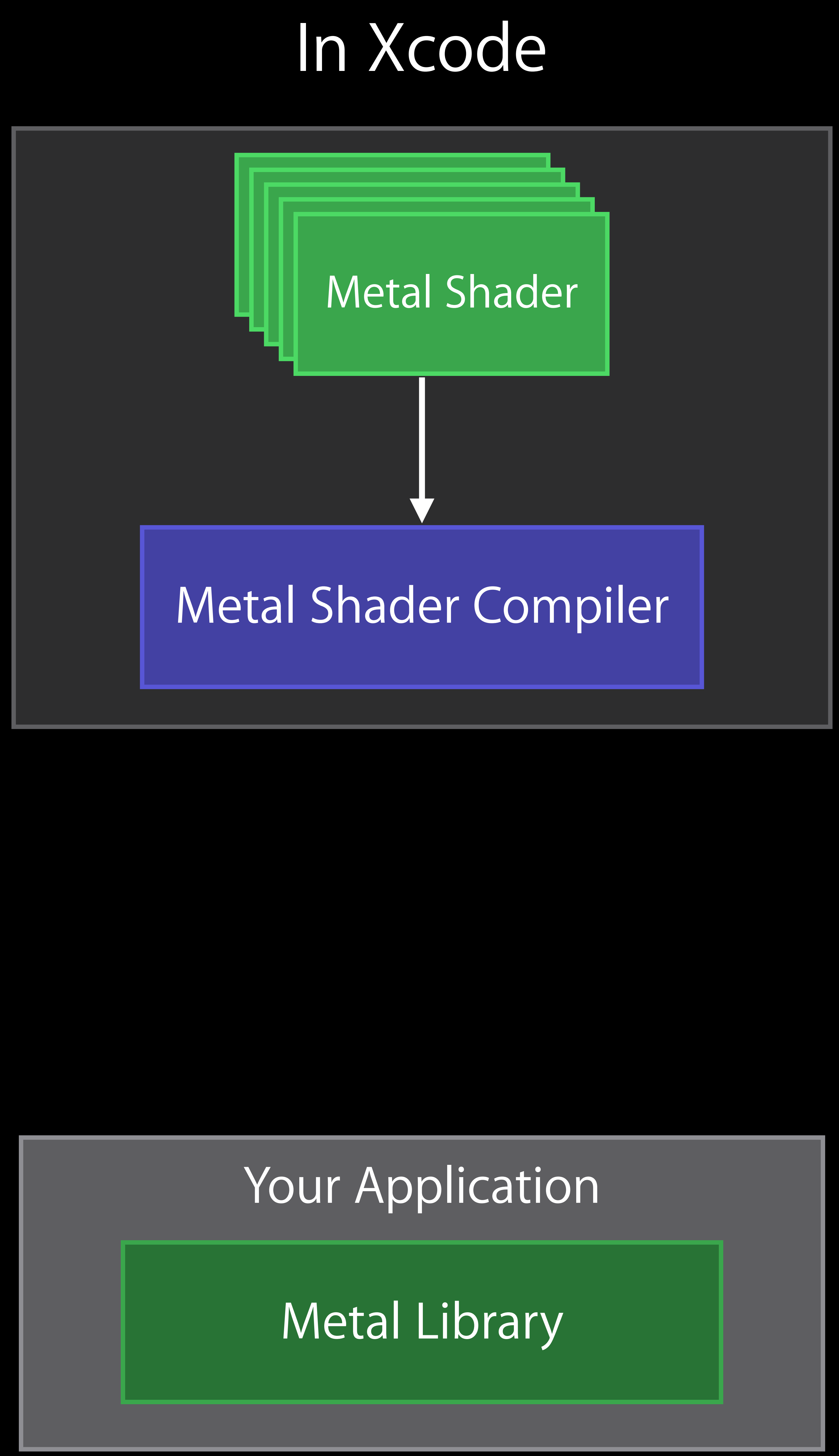
Metal Shader Compiler Process



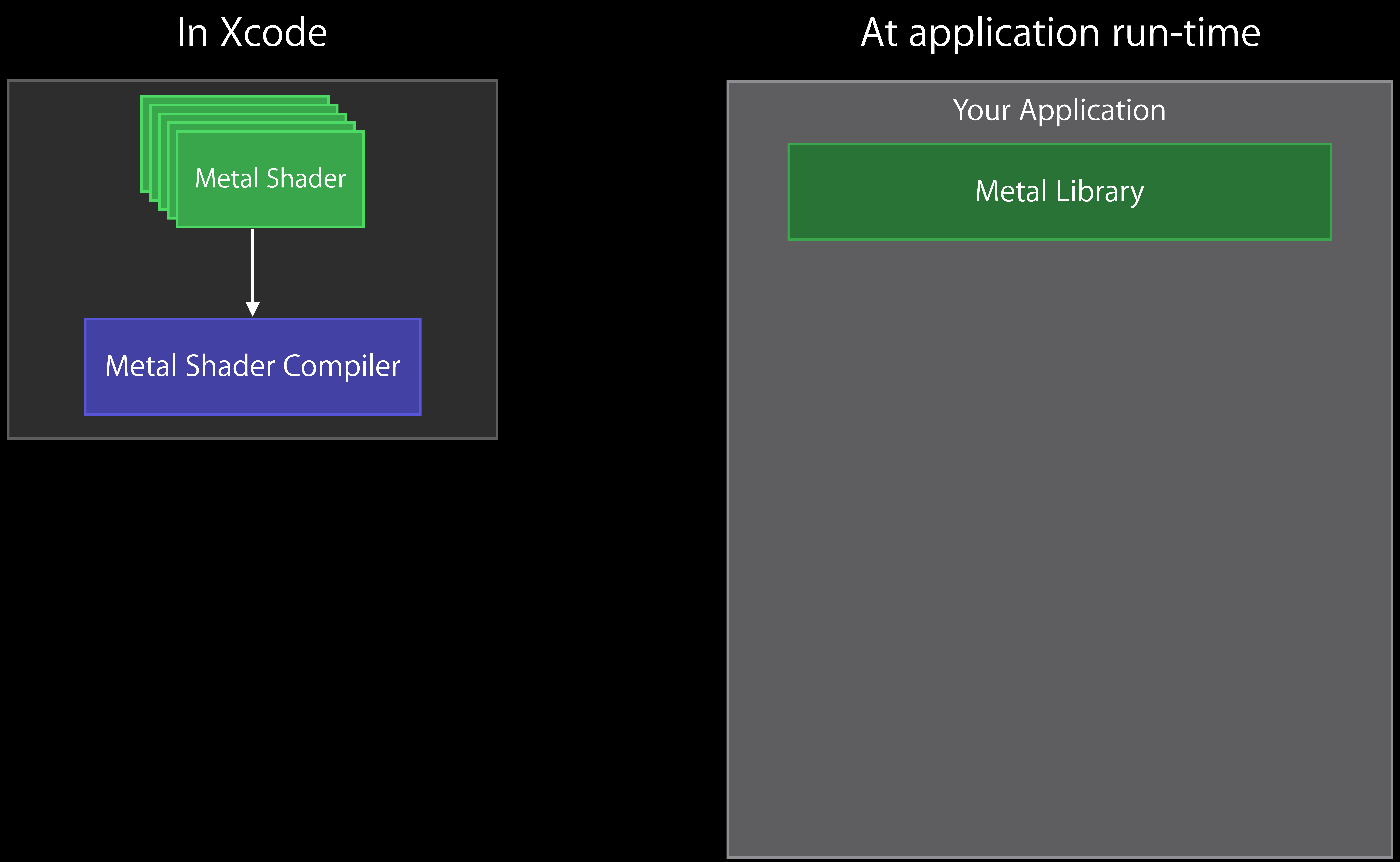
Metal Shader Compiler Process



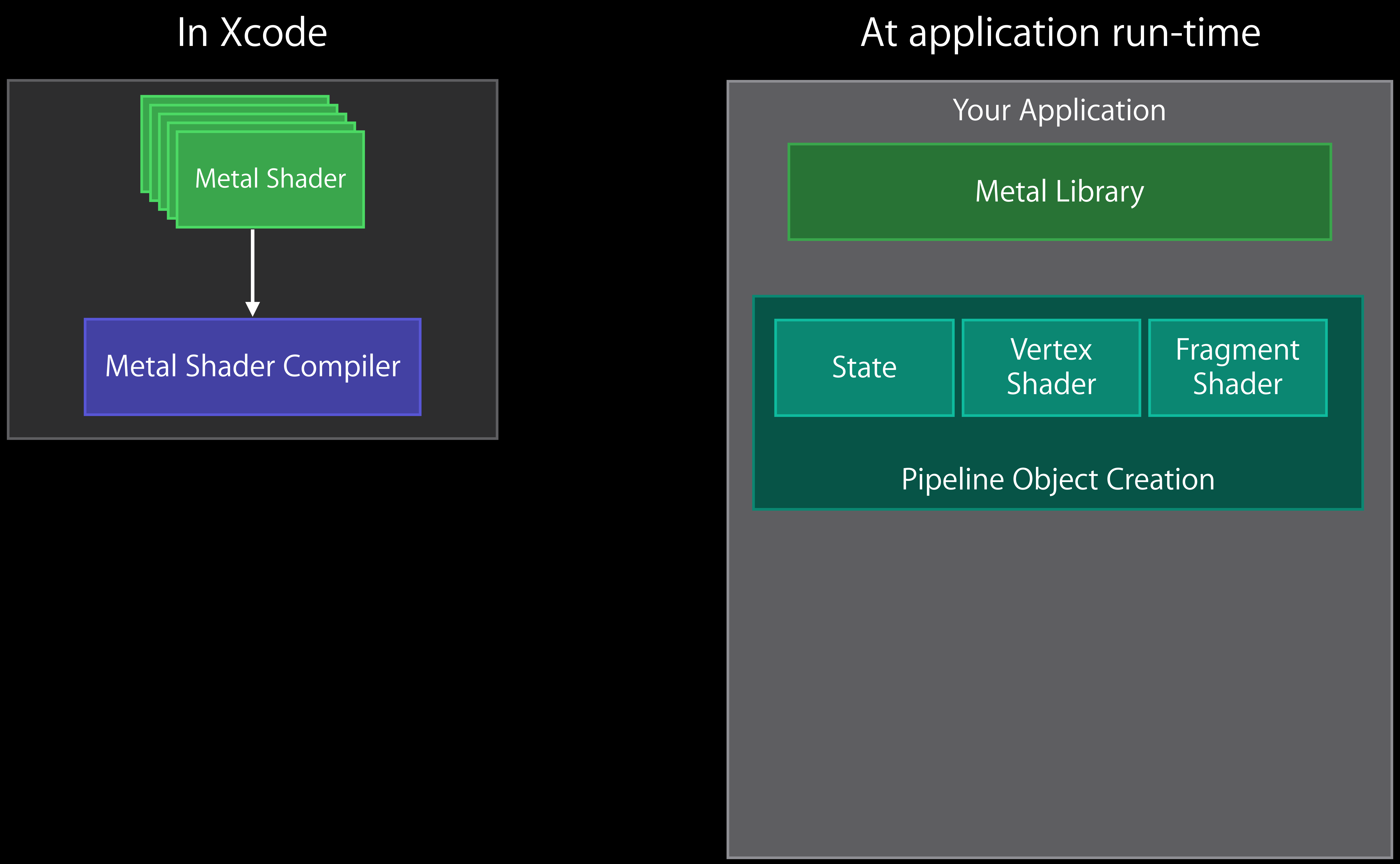
Metal Shader Compiler Process



Metal Shader Compiler Process

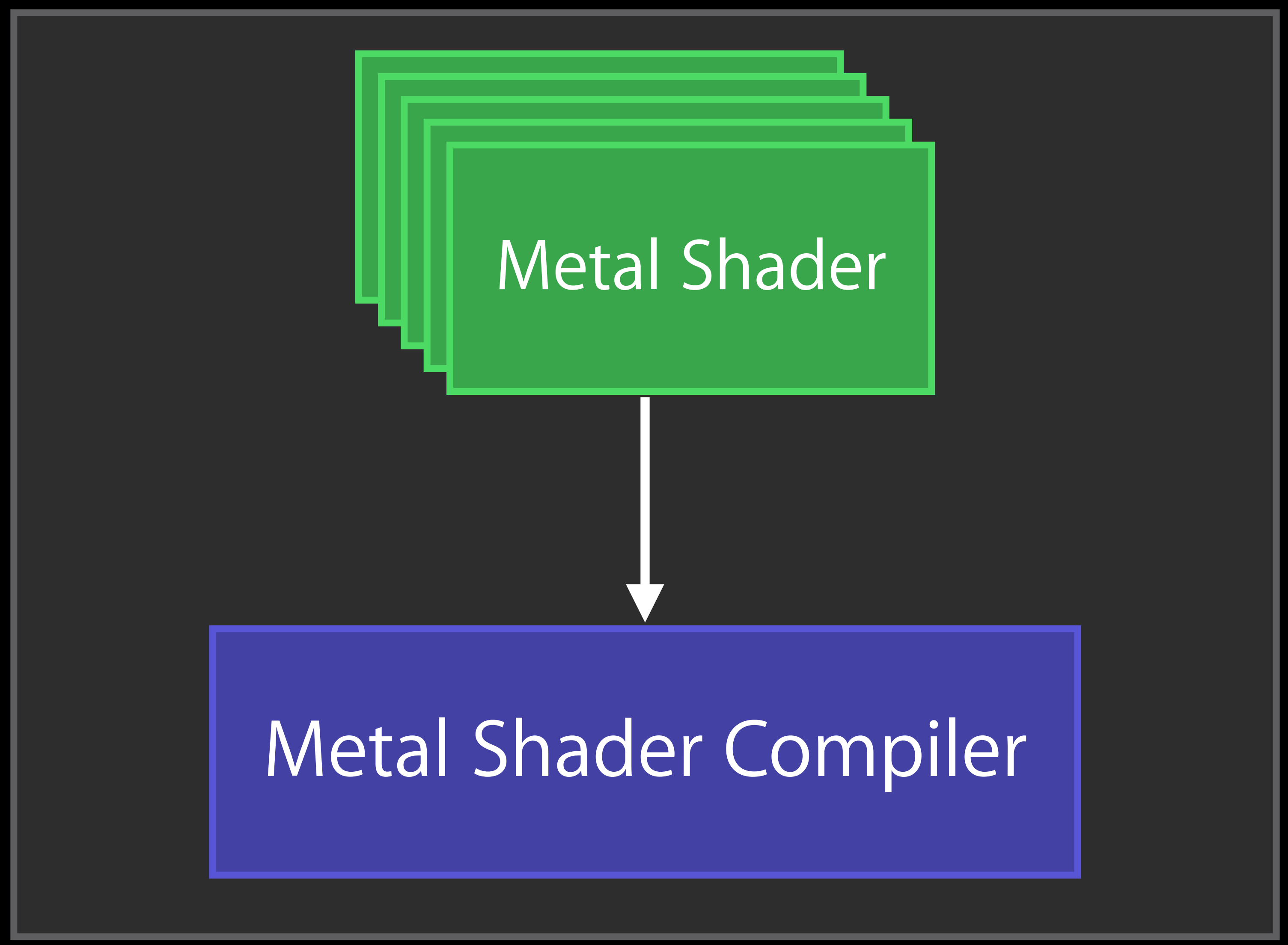


Metal Shader Compiler Process

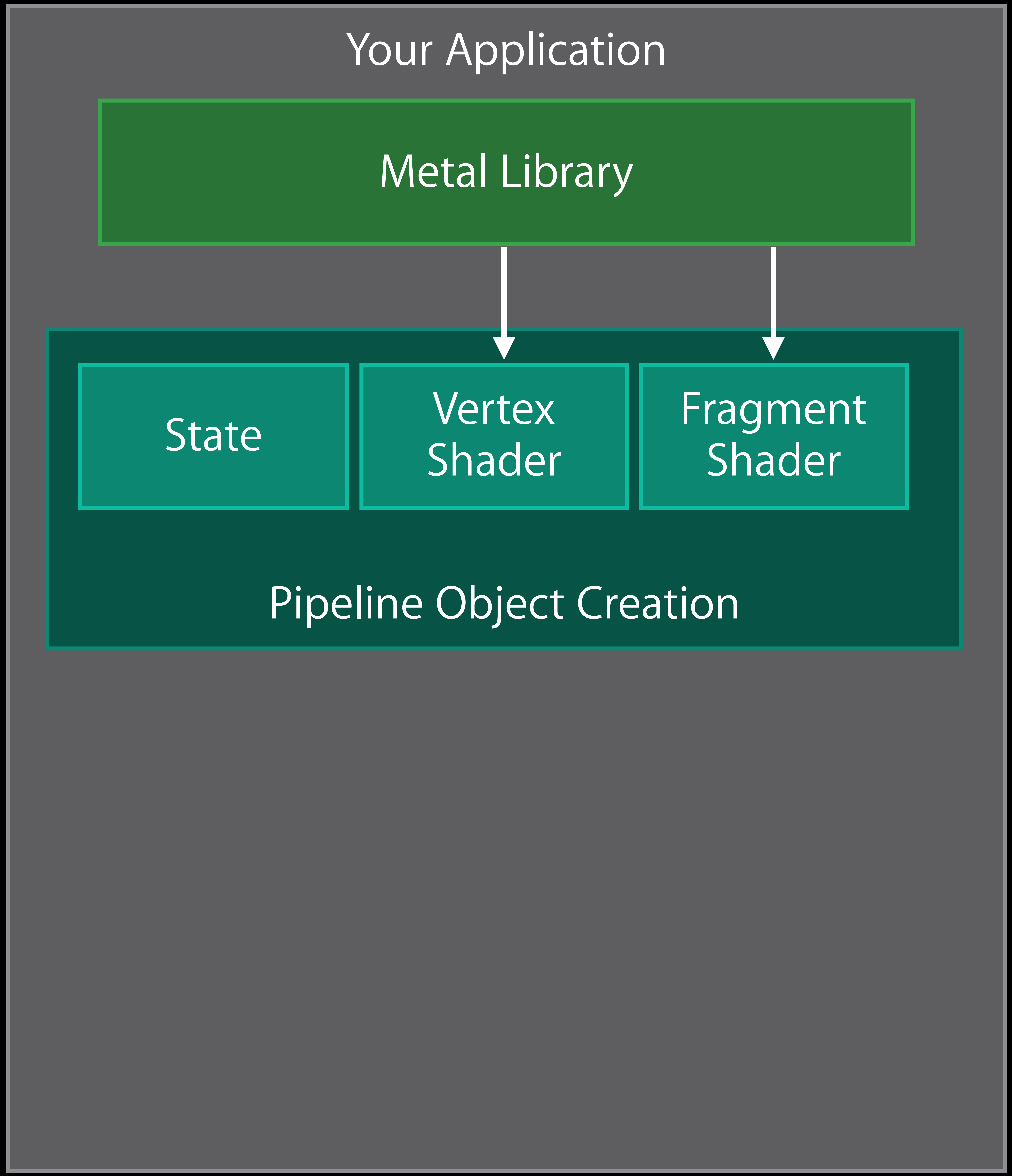


Metal Shader Compiler Process

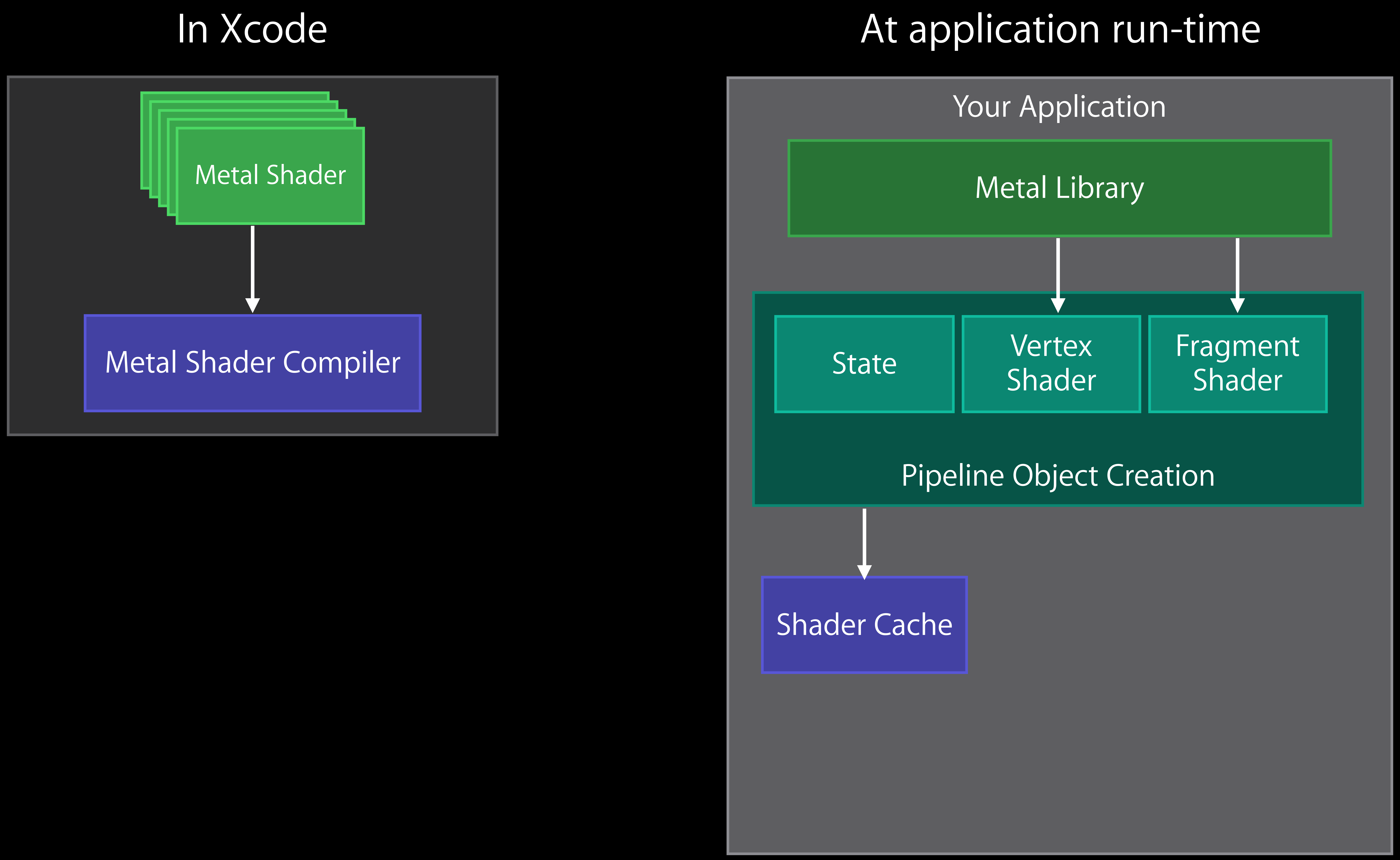
In Xcode



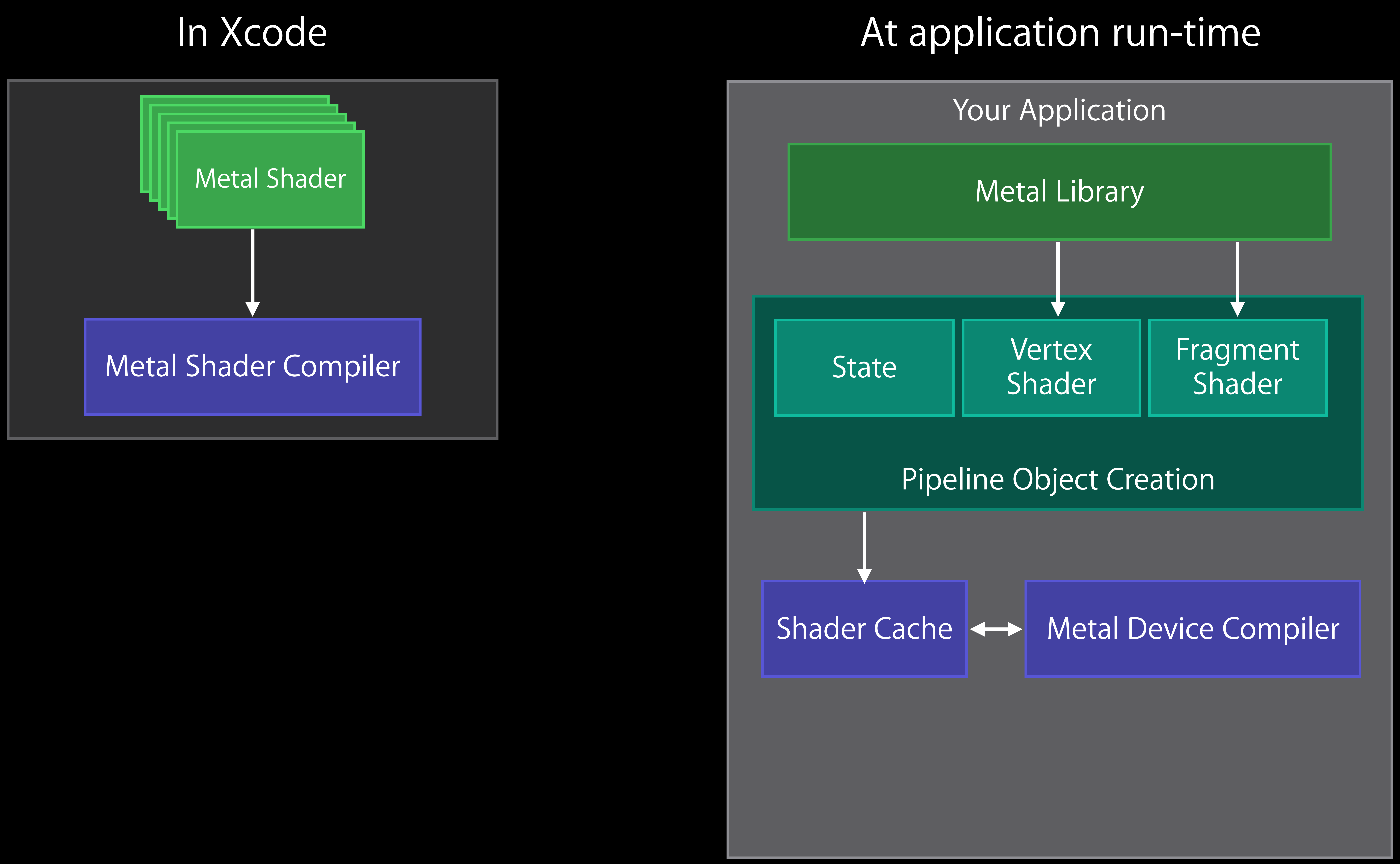
At application run-time



Metal Shader Compiler Process

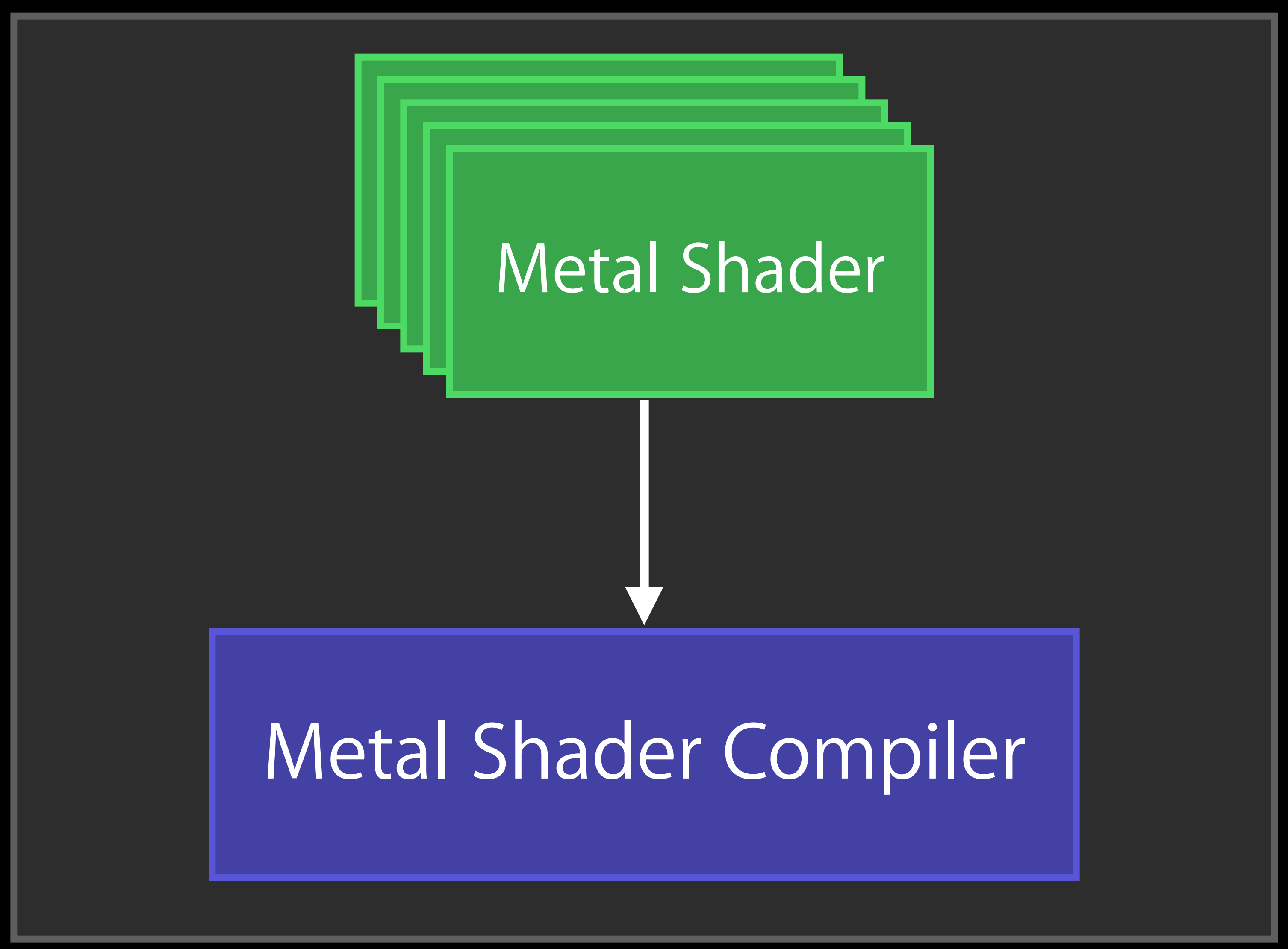


Metal Shader Compiler Process

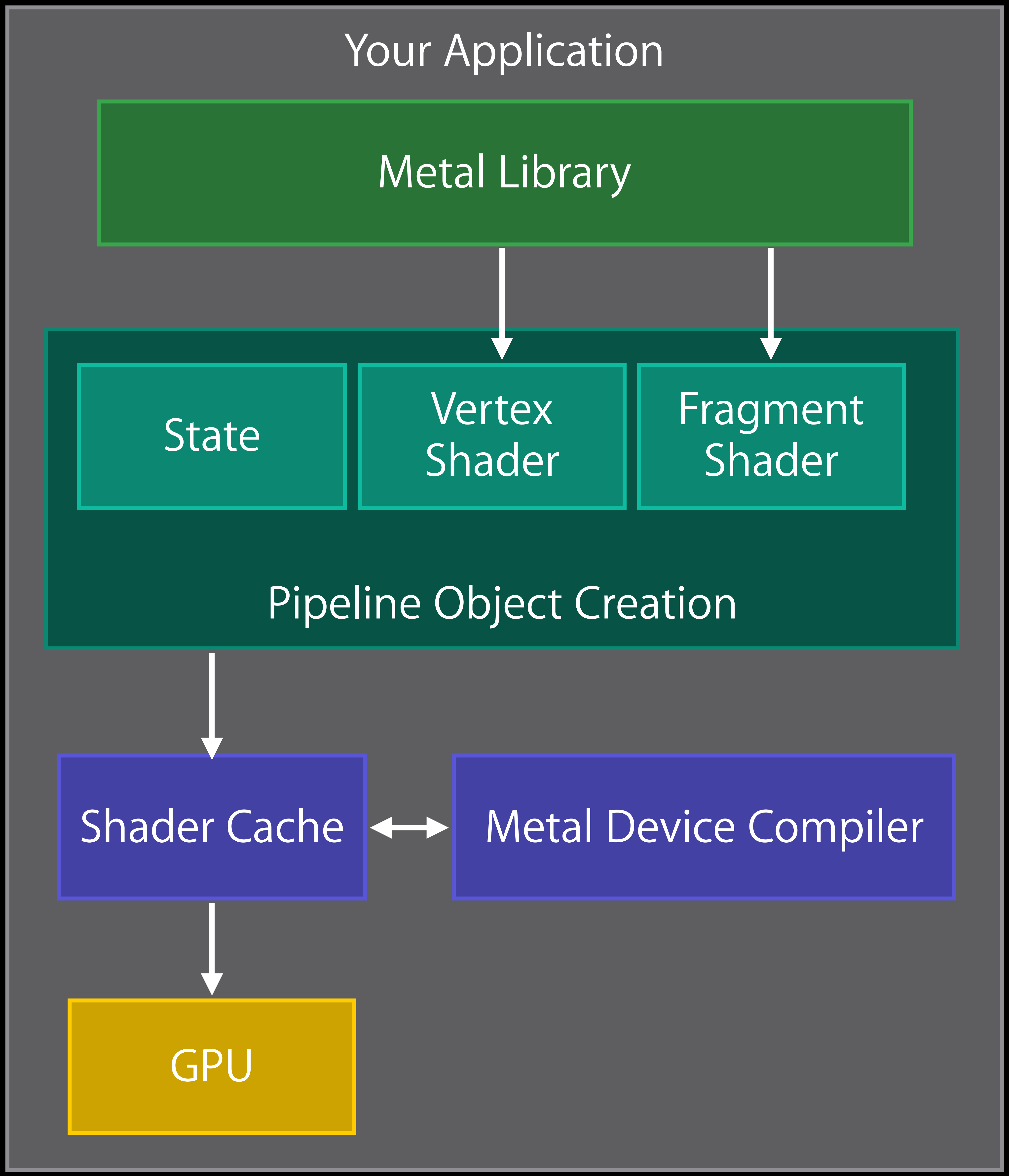


Metal Shader Compiler Process

In Xcode

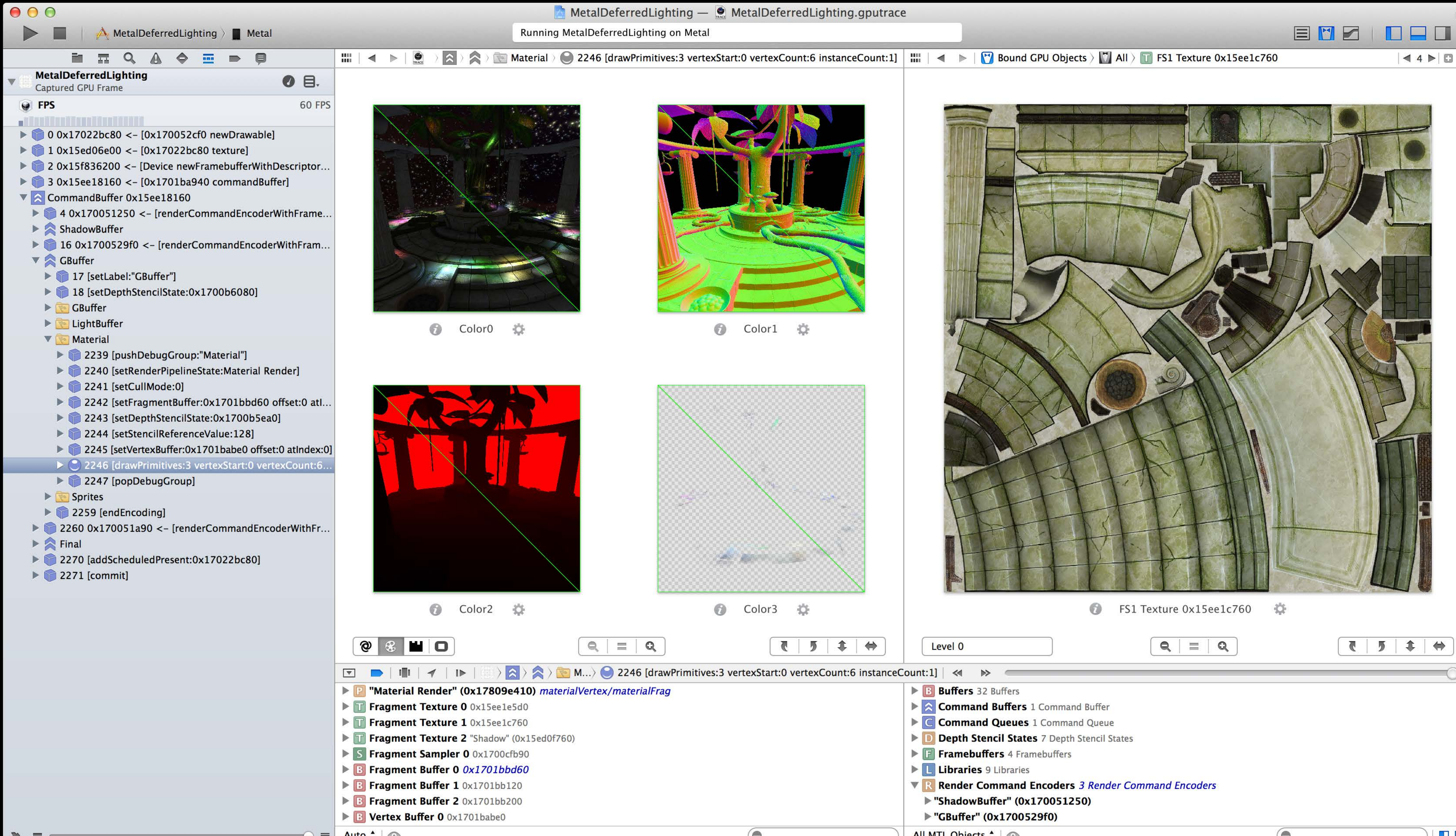


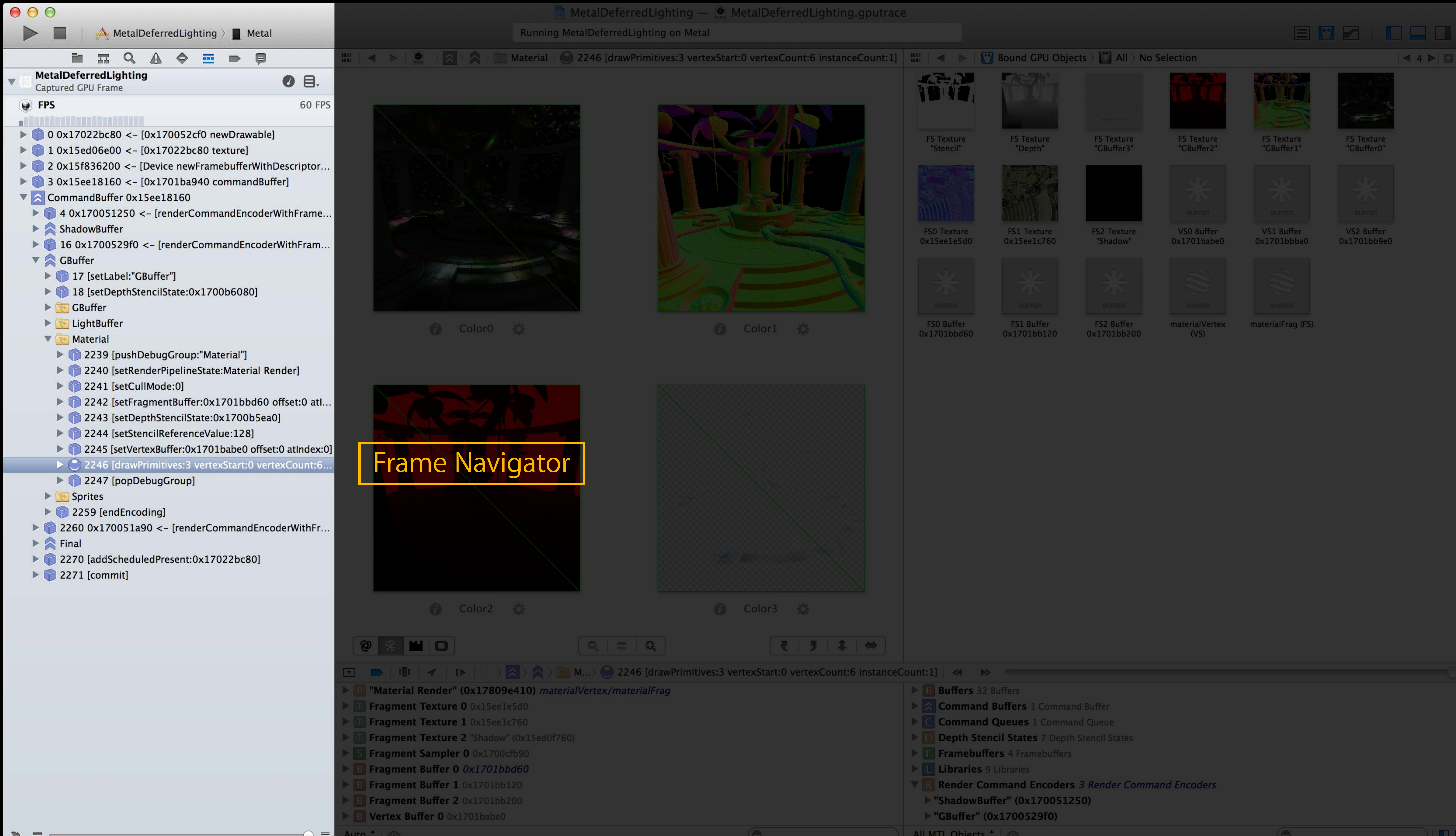
At application run-time

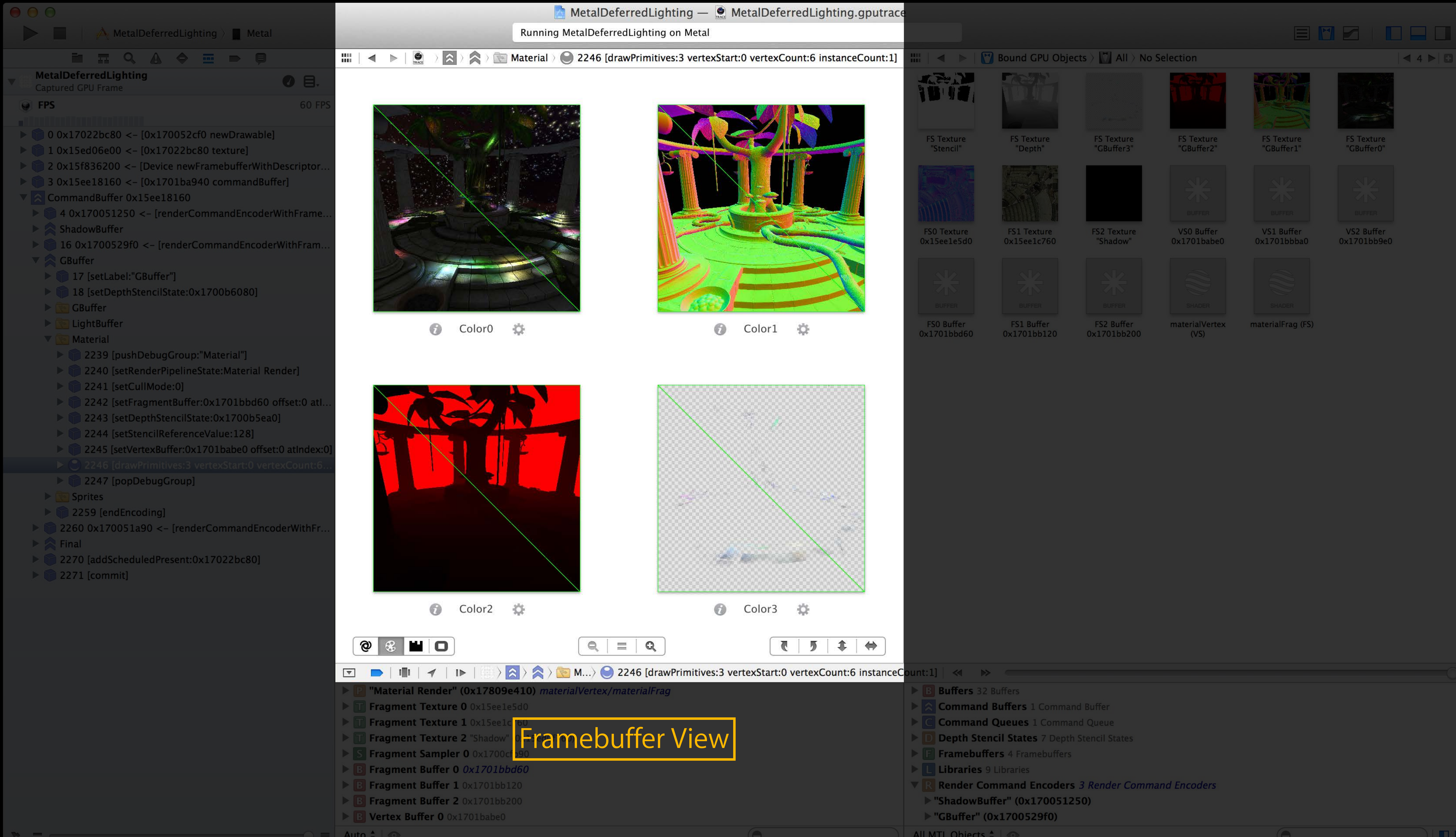


Metal Tools Fully Integrated into Xcode

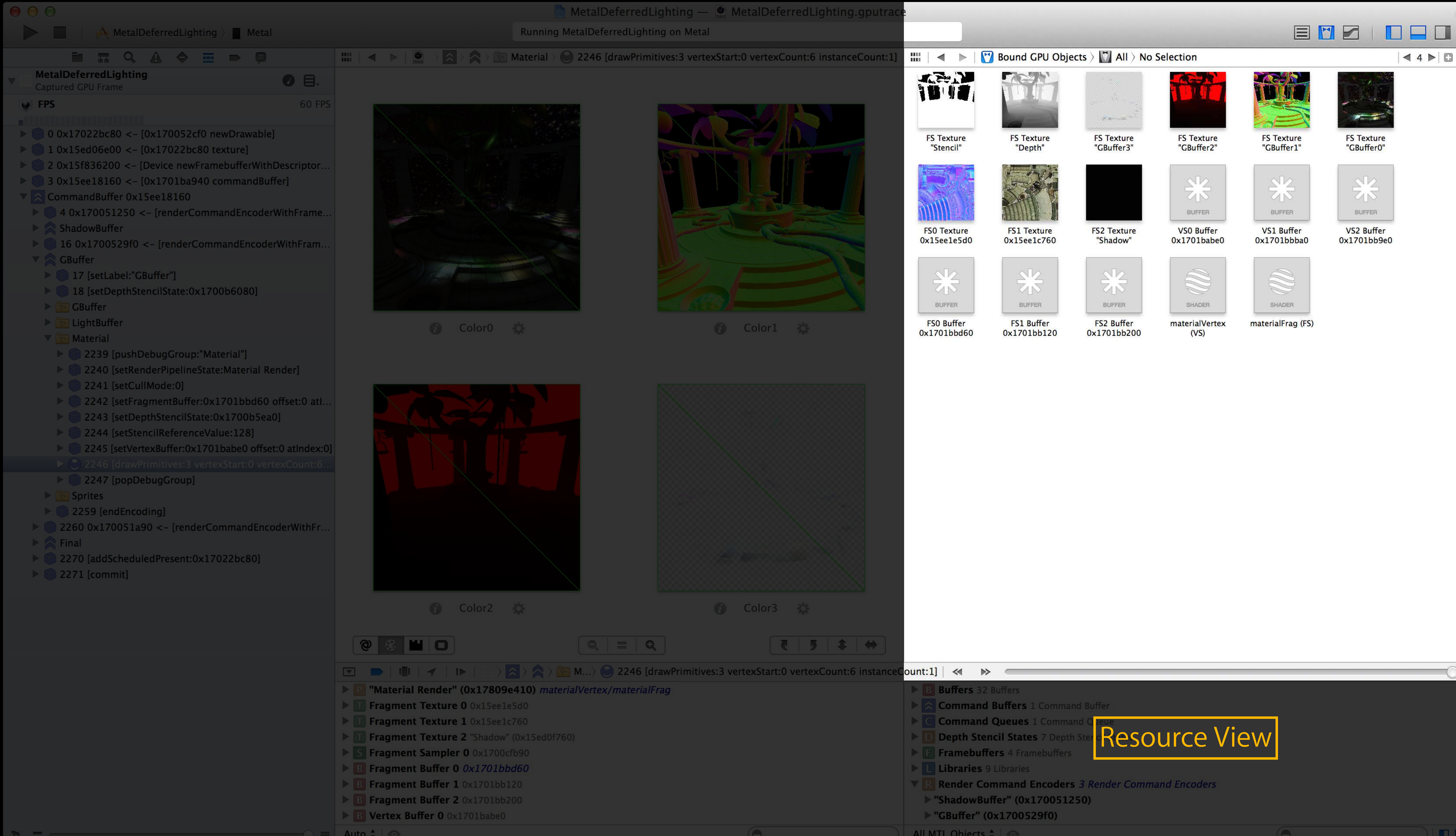
- Visual frame debugger
- API trace and navigation
- Shader edit-and-continue
- Rich source code editing (including shaders)
- Graphics and compute state inspection
- Shader compiler
- Debug mode for Metal framework

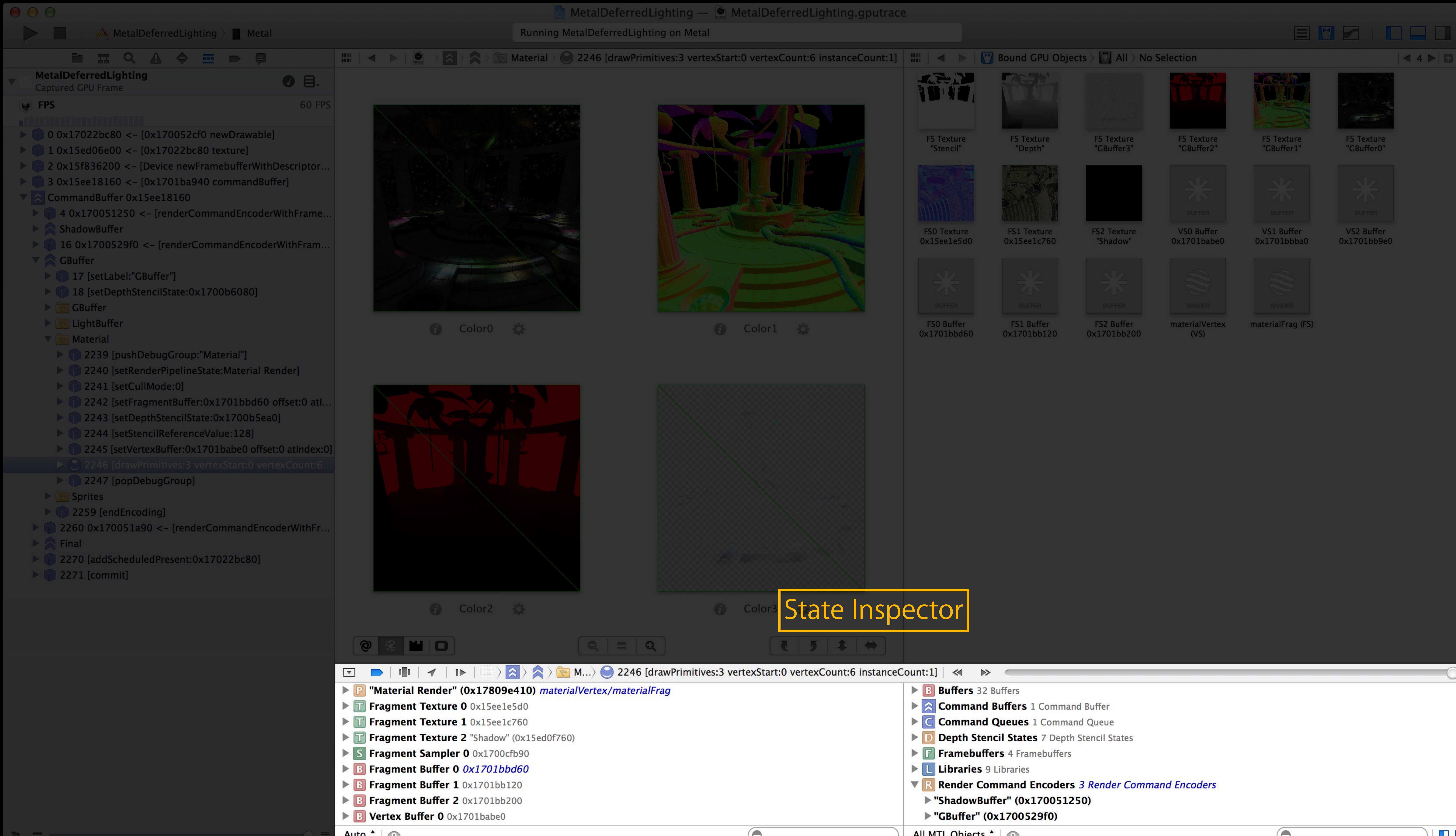

MetalDeferredLighting — MetalDeferredLighting.gputrace
MetalDeferredLighting
Metal
Running MetalDeferredLighting on Metal
Material
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6 instanceCount:1]
Bound GPU Objects
All
FS1 Texture 0x15ee1c760
MetalDeferredLighting
Captured GPU Frame
FPS
60 FPS
0 0x17022bc80 <- [0x170052cf0 newDrawable]
1 0x15ed06e00 <- [0x17022bc80 texture]
2 0x15f836200 <- [Device newFramebufferWithDescriptor…
3 0x15ee18160 <- [0x1701ba940 commandBuffer]
CommandBuffer 0x15ee18160
4 0x170051250 <- [renderCommandEncoderWithFrame…
ShadowBuffer
16 0x1700529f0 <- [renderCommandEncoderWithFram…
GBuffer
17 [setLabel:"GBuffer"]
18 [setDepthStencilState:0x1700b6080]
GBuffer
LightBuffer
Material
2239 [pushDebugGroup:"Material"]
2240 [setRenderPipelineState:Material Render]
2241 [setCullMode:0]
2242 [setFragmentBuffer:0x1701bbd60 offset:0 atI…
2243 [setDepthStencilState:0x1700b5ea0]
2244 [setStencilReferenceValue:128]
2245 [setVertexBuffer:0x1701babe0 offset:0 atIndex:0]
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6…
2247 [popDebugGroup]
Sprites
2259 [endEncoding]
2260 0x170051a90 <- [renderCommandEncoderWithFr…
Final
2270 [addScheduledPresent:0x17022bc80]
2271 [commit]
Color0
Color1
Color2
Color3
FS1 Texture 0x15ee1c760
Level 0
M…
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6 instanceCount:1]
"Material Render" (0x17809e410) materialVertex/materialFrag
Fragment Texture 0 0x15ee1e5d0
Fragment Texture 1 0x15ee1c760
Fragment Texture 2 "Shadow" (0x15ed0f760)
Fragment Sampler 0 0x1700cfb90
Fragment Buffer 0 0x1701bbd60
Fragment Buffer 1 0x1701bb120
Fragment Buffer 2 0x1701bb200
Vertex Buffer 0 0x1701babe0
Buffers 32 Buffers
Command Buffers 1 Command Buffer
Command Queues 1 Command Queue
Depth Stencil States 7 Depth Stencil States
Framebuffers 4 Framebuffers
Libraries 9 Libraries
Render Command Encoders 3 Render Command Encoders
"ShadowBuffer" (0x170051250)
"GBuffer" (0x1700529f0)
Auto
All MTL Objects

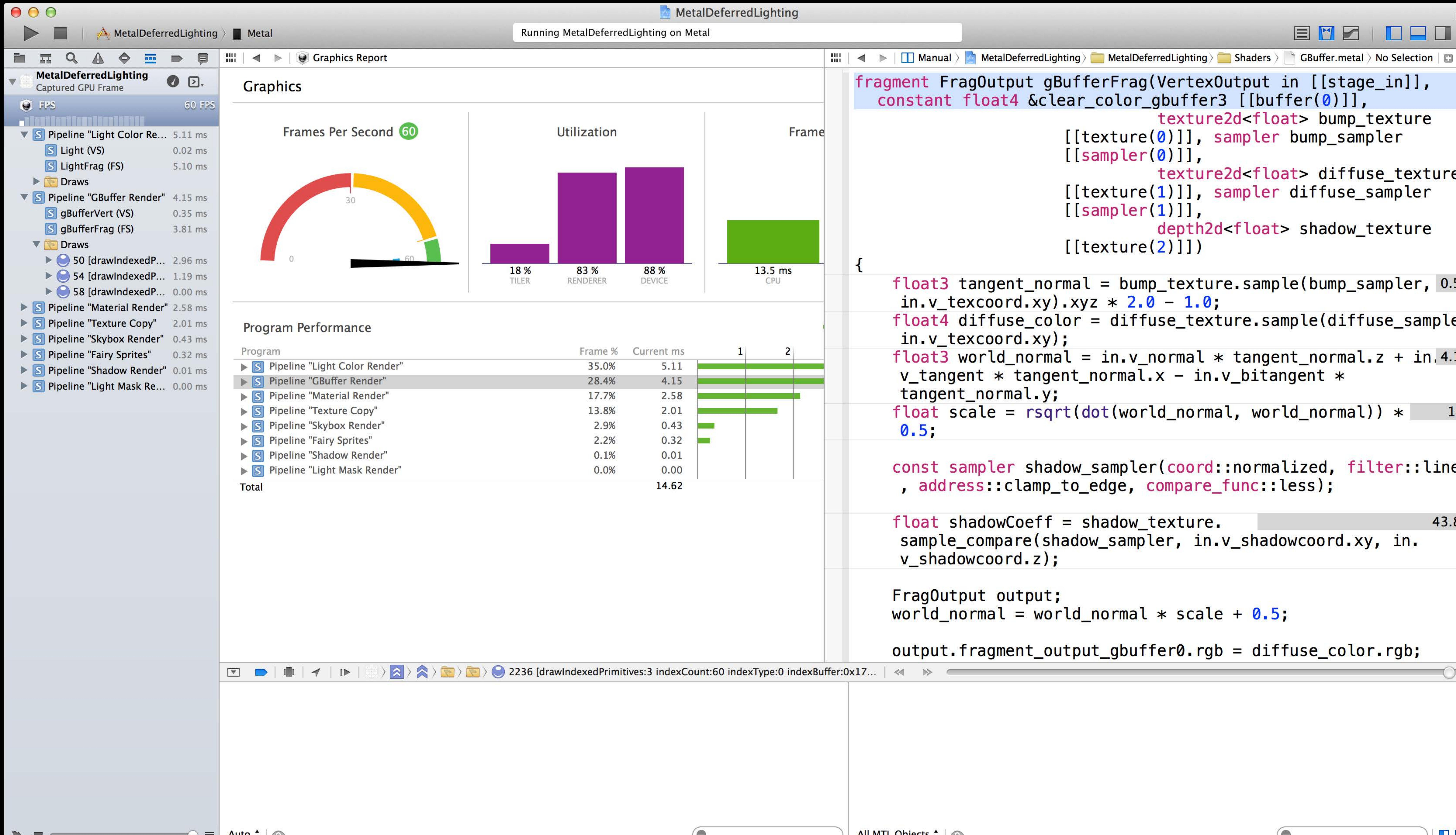

MetalDeferredLighting — MetalDeferredLighting.gputrace
MetalDeferredLighting
Metal
Running MetalDeferredLighting on Metal
MetalDeferredLighting
Captured GPU Frame
FPS
60 FPS
0 0x17022bc80 <- [0x170052cf0 newDrawable]
1 0x15ed06e00 <- [0x17022bc80 texture]
2 0x15f836200 <- [Device newFramebufferWithDescriptor…
3 0x15ee18160 <- [0x1701ba940 commandBuffer]
CommandBuffer 0x15ee18160
4 0x170051250 <- [renderCommandEncoderWithFrame…
ShadowBuffer
16 0x1700529f0 <- [renderCommandEncoderWithFram…
GBuffer
17 [setLabel:"GBuffer"]
18 [setDepthStencilState:0x1700b6080]
GBuffer
LightBuffer
Material
2239 [pushDebugGroup:"Material"]
2240 [setRenderPipelineState:Material Render]
2241 [setCullMode:0]
2242 [setFragmentBuffer:0x1701bbd60 offset:0 atI…
2243 [setDepthStencilState:0x1700b5ea0]
2244 [setStencilReferenceValue:128]
2245 [setVertexBuffer:0x1701babe0 offset:0 atIndex:0]
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6…
2247 [popDebugGroup]
Sprites
2259 [endEncoding]
2260 0x170051a90 <- [renderCommandEncoderWithFr…
Final
2270 [addScheduledPresent:0x17022bc80]
2271 [commit]
Material
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6 instanceCount:1]
Bound GPU Objects
All
No Selection
Color0
Color1
Color2
Color3
Frame Navigator
FS Texture "Stencil"
FS Texture "Depth"
FS Texture "GBuffer3"
FS Texture "GBuffer2"
FS Texture "GBuffer1"
FS Texture "GBuffer0"
FS0 Texture 0x15ee1e5d0
FS1 Texture 0x15ee1c760
FS2 Texture "Shadow"
VS0 Buffer 0x1701babe0
VS1 Buffer 0x1701bbba0
VS2 Buffer 0x1701bb9e0
FS0 Buffer 0x1701bbd60
FS1 Buffer 0x1701bb120
FS2 Buffer 0x1701bb200
materialVertex (VS)
materialFrag (FS)
"Material Render" (0x17809e410) materialVertex/materialFrag
Fragment Texture 0 0x15ee1e5d0
Fragment Texture 1 0x15ee1c760
Fragment Texture 2 "Shadow" (0x15ed0f760)
Fragment Sampler 0 0x1700cfb90
Fragment Buffer 0 0x1701bbd60
Fragment Buffer 1 0x1701bb120
Fragment Buffer 2 0x1701bb200
Vertex Buffer 0 0x1701babe0
Buffers 32 Buffers
Command Buffers 1 Command Buffer
Command Queues 1 Command Queue
Depth Stencil States 7 Depth Stencil States
Framebuffers 4 Framebuffers
Libraries 9 Libraries
Render Command Encoders 3 Render Command Encoders
"ShadowBuffer" (0x170051250)
"GBuffer" (0x1700529f0)
Auto
All MTL Objects

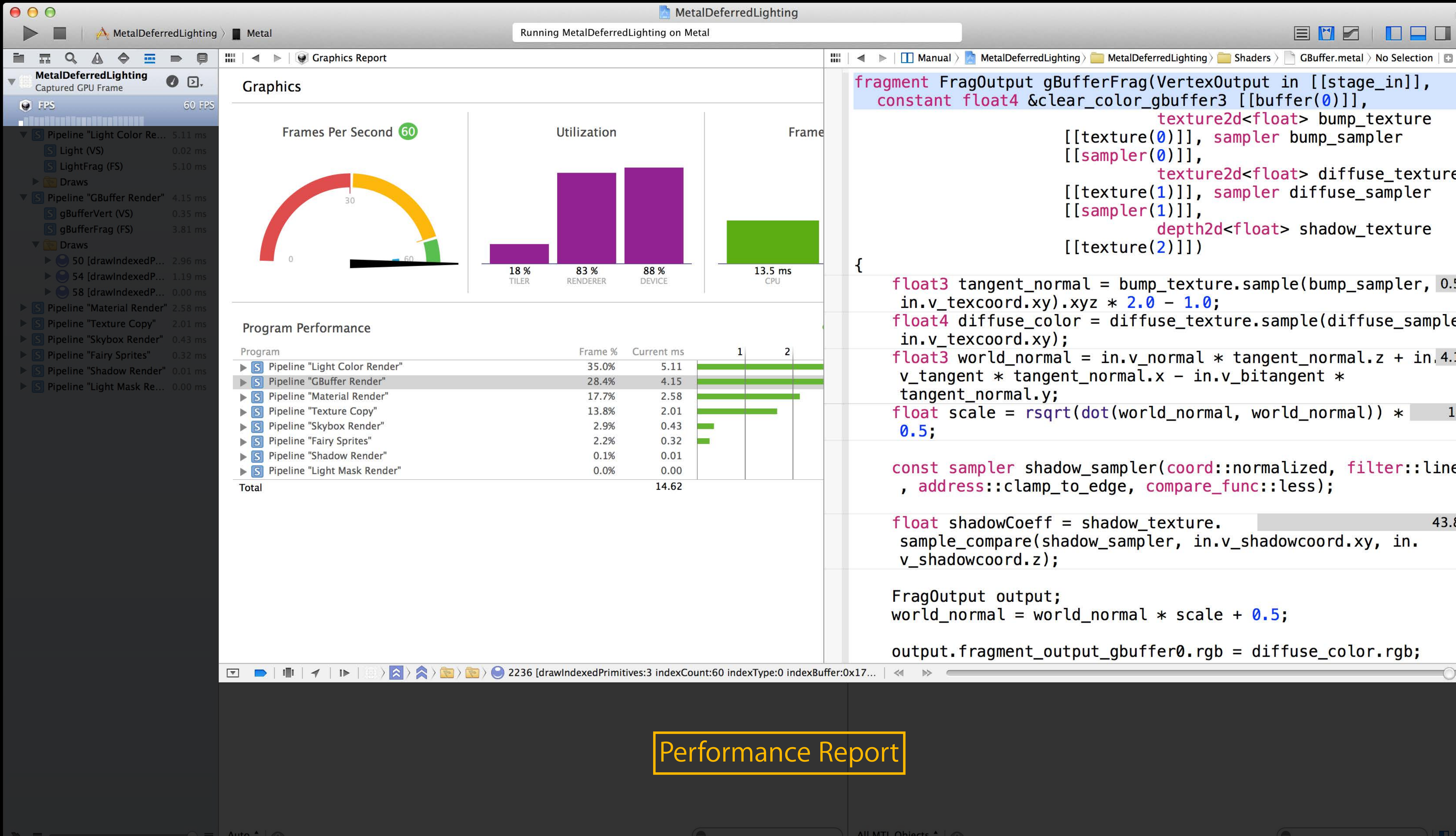


MetalDeferredLighting — MetalDeferredLighting.gputrace
Running MetalDeferredLighting on Metal
MetalDeferredLighting
Captured GPU Frame
FPS
60 FPS
0 0x17022bc80 <- [0x170052cf0 newDrawable]
1 0x15ed06e00 <- [0x17022bc80 texture]
2 0x15f836200 <- [Device newFramebufferWithDescriptor…
3 0x15ee18160 <- [0x1701ba940 commandBuffer]
CommandBuffer 0x15ee18160
4 0x170051250 <- [renderCommandEncoderWithFrame…
ShadowBuffer
16 0x1700529f0 <- [renderCommandEncoderWithFram…
GBuffer
17 [setLabel:"GBuffer"]
18 [setDepthStencilState:0x1700b6080]
GBuffer
LightBuffer
Material
2239 [pushDebugGroup:"Material"]
2240 [setRenderPipelineState:Material Render]
2241 [setCullMode:0]
2242 [setFragmentBuffer:0x1701bbd60 offset:0 atl…
2243 [setDepthStencilState:0x1700b5ea0]
2244 [setStencilReferenceValue:128]
2245 [setVertexBuffer:0x1701babe0 offset:0 atIndex:0]
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6…
2247 [popDebugGroup]
Sprites
2259 [endEncoding]
2260 0x170051a90 <- [renderCommandEncoderWithFr…
Final
2270 [addScheduledPresent:0x17022bc80]
2271 [commit]
Material
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6 instanceCount:1]
Color0
Color1
Color2
Color3
Bound GPU Objects
All
No Selection
FS Texture "Stencil"
FS Texture "Depth"
FS Texture "GBuffer3"
FS Texture "GBuffer2"
FS Texture "GBuffer1"
FS Texture "GBuffer0"
FS0 Texture 0x15ee1e5d0
FS1 Texture 0x15ee1c760
FS2 Texture "Shadow"
VS0 Buffer 0x1701babe0
VS1 Buffer 0x1701bbba0
VS2 Buffer 0x1701bb9e0
FS0 Buffer 0x1701bbd60
FS1 Buffer 0x1701bb120
FS2 Buffer 0x1701bb200
materialVertex (VS)
materialFrag (FS)
"Material Render" (0x17809e410) materialVertex/materialFrag
Fragment Texture 0 0x15ee1e5d0
Fragment Texture 1
Fragment Texture 2 "Shadow"
Fragment Sampler 0
Fragment Buffer 0 0x1701bbd60
Fragment Buffer 1 0x1701bb120
Fragment Buffer 2 0x1701bb200
Vertex Buffer 0 0x1701babe0
Framebuffer View
Buffers 32 Buffers
Command Buffers 1 Command Buffer
Command Queues 1 Command Queue
Depth Stencil States 7 Depth Stencil States
Framebuffers 4 Framebuffers
Libraries 9 Libraries
Render Command Encoders 3 Render Command Encoders
"ShadowBuffer" (0x170051250)
"GBuffer" (0x1700529f0)
Auto
All MTL Objects

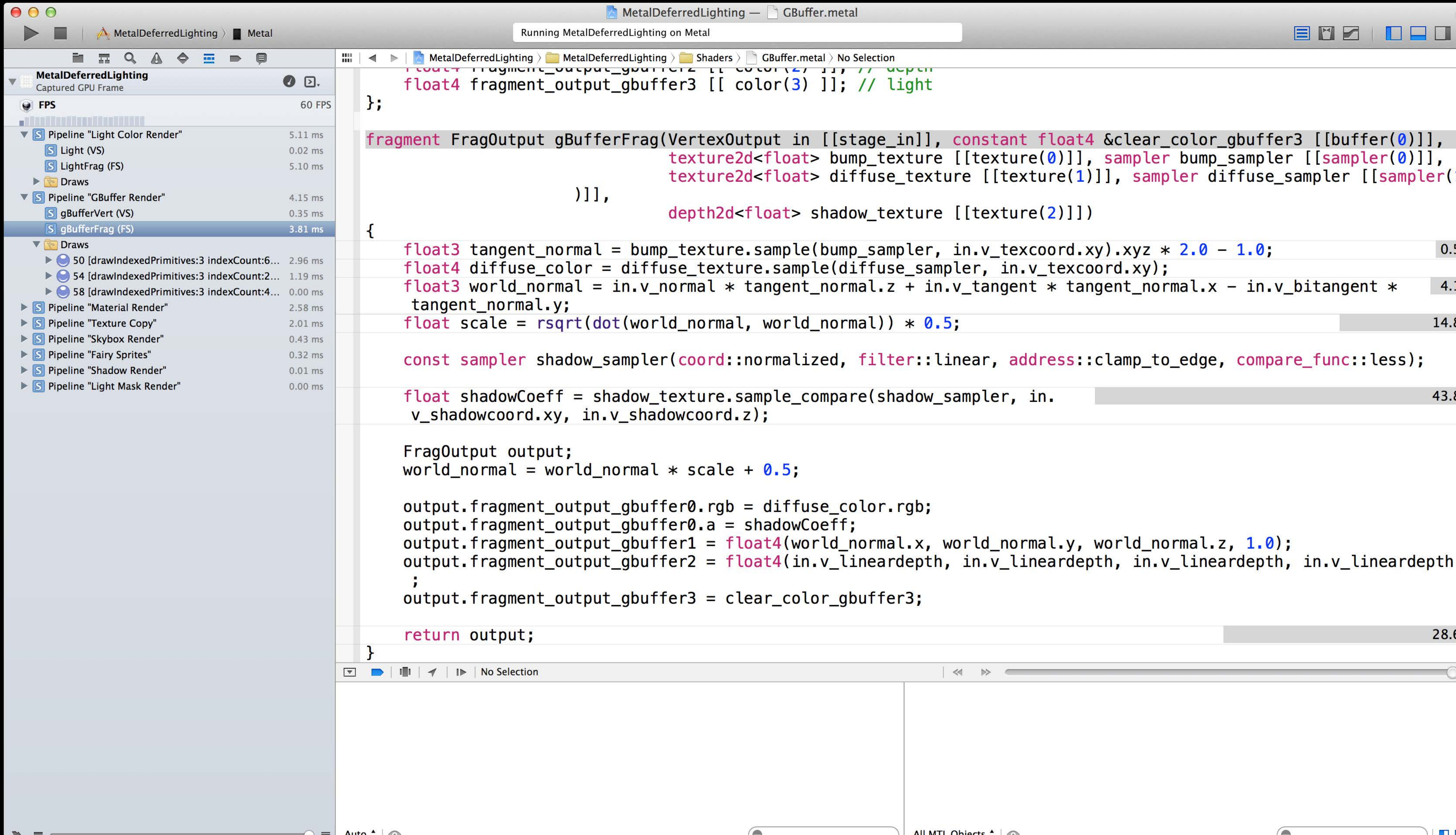

MetalDeferredLighting — MetalDeferredLighting.gputrace
MetalDeferredLighting
Metal
Running MetalDeferredLighting on Metal
MetalDeferredLighting
Captured GPU Frame
FPS
60 FPS
0 0x17022bc80 <- [0x170052cf0 newDrawable]
1 0x15ed06e00 <- [0x17022bc80 texture]
2 0x15f836200 <- [Device newFramebufferWithDescriptor...
3 0x15ee18160 <- [0x1701ba940 commandBuffer]
CommandBuffer 0x15ee18160
4 0x170051250 <- [renderCommandEncoderWithFrame...
ShadowBuffer
16 0x1700529f0 <- [renderCommandEncoderWithFram...
GBuffer
17 [setLabel:"GBuffer"]
18 [setDepthStencilState:0x1700b6080]
GBuffer
LightBuffer
Material
2239 [pushDebugGroup:"Material"]
2240 [setRenderPipelineState:Material Render]
2241 [setCullMode:0]
2242 [setFragmentBuffer:0x1701bbd60 offset:0 atI...
2243 [setDepthStencilState:0x1700b5ea0]
2244 [setStencilReferenceValue:128]
2245 [setVertexBuffer:0x1701babe0 offset:0 atIndex:0]
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6...
2247 [popDebugGroup]
Sprites
2259 [endEncoding]
2260 0x170051a90 <- [renderCommandEncoderWithFr...
Final
2270 [addScheduledPresent:0x17022bc80]
2271 [commit]
Material
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6 instanceCount:1]
Color0
Color1
Color2
Color3
"Material Render" (0x17809e410) materialVertex/materialFrag
Fragment Texture 0 0x15ee1e5d0
Fragment Texture 1 0x15ee1c760
Fragment Texture 2 "Shadow" (0x15ed0f760)
Fragment Sampler 0 0x1700cfb90
Fragment Buffer 0 0x1701bbd60
Fragment Buffer 1 0x1701bb120
Fragment Buffer 2 0x1701bb200
Vertex Buffer 0 0x1701babe0
Auto
Bound GPU Objects
All
No Selection
FS Texture "Stencil"
FS Texture "Depth"
FS Texture "GBuffer3"
FS Texture "GBuffer2"
FS Texture "GBuffer1"
FS Texture "GBuffer0"
FS0 Texture 0x15ee1e5d0
FS1 Texture 0x15ee1c760
FS2 Texture "Shadow"
VS0 Buffer 0x1701babe0
VS1 Buffer 0x1701bbba0
VS2 Buffer 0x1701bb9e0
FS0 Buffer 0x1701bbd60
FS1 Buffer 0x1701bb120
FS2 Buffer 0x1701bb200
materialVertex (VS)
materialFrag (FS)
Buffers 32 Buffers
Command Buffers 1 Command Buffer
Command Queues 1 Command Queue
Depth Stencil States 7 Depth Ste...
Framebuffers 4 Framebuffers
Libraries 9 Libraries
Render Command Encoders 3 Render Command Encoders
"ShadowBuffer" (0x170051250)
"GBuffer" (0x1700529f0)
All MTL Objects
Resource View

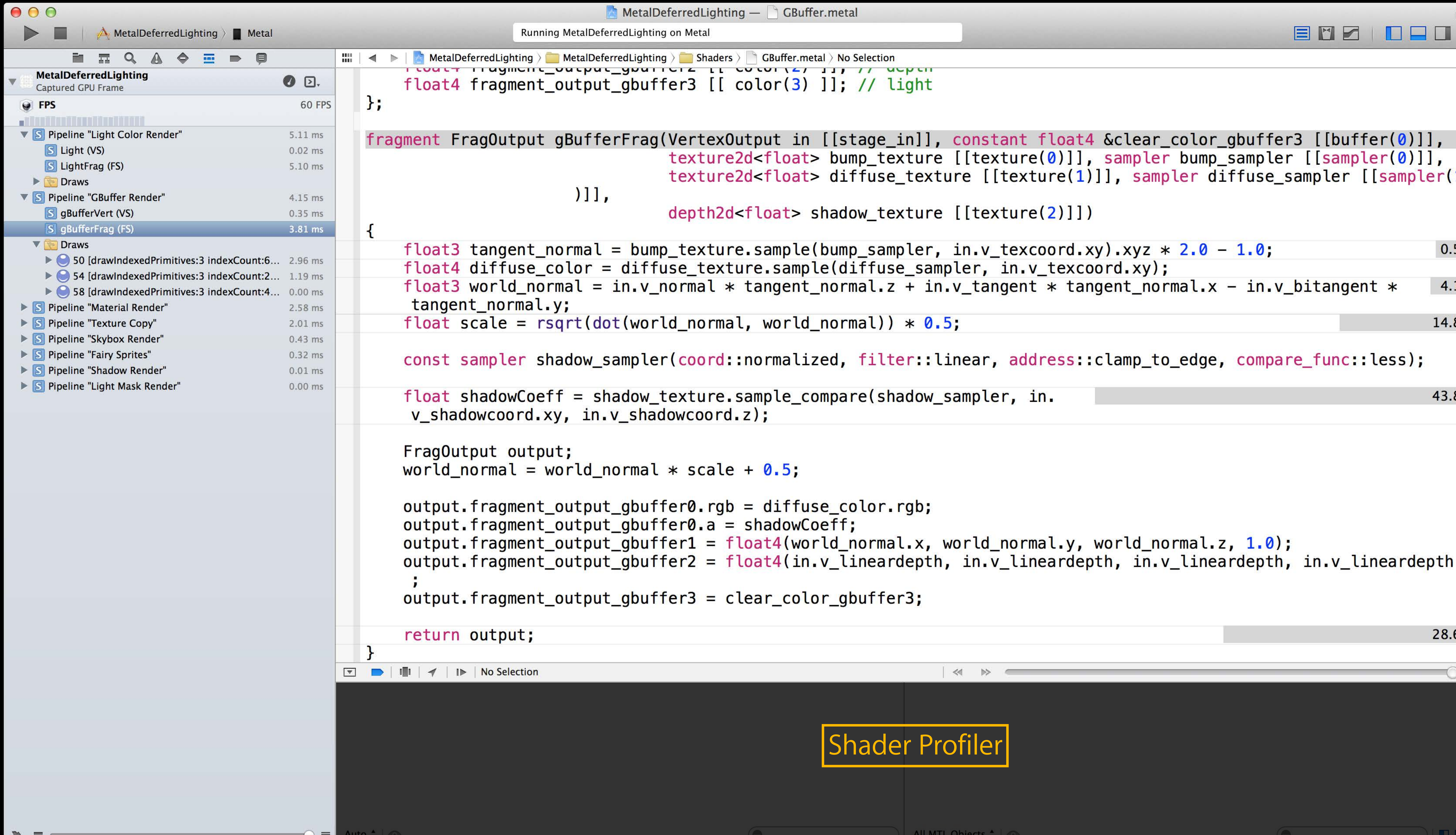

MetalDeferredLighting — MetalDeferredLighting.gputrace
MetalDeferredLighting
Metal
Running MetalDeferredLighting on Metal
MetalDeferredLighting
Captured GPU Frame
FPS
60 FPS
0 0x17022bc80 <- [0x170052cf0 newDrawable]
1 0x15ed06e00 <- [0x17022bc80 texture]
2 0x15f836200 <- [Device newFramebufferWithDescriptor…
3 0x15ee18160 <- [0x1701ba940 commandBuffer]
CommandBuffer 0x15ee18160
4 0x170051250 <- [renderCommandEncoderWithFrame…
ShadowBuffer
16 0x1700529f0 <- [renderCommandEncoderWithFram…
GBuffer
17 [setLabel:"GBuffer"]
18 [setDepthStencilState:0x1700b6080]
GBuffer
LightBuffer
Material
2239 [pushDebugGroup:"Material"]
2240 [setRenderPipelineState:Material Render]
2241 [setCullMode:0]
2242 [setFragmentBuffer:0x1701bbd60 offset:0 atI…
2243 [setDepthStencilState:0x1700b5ea0]
2244 [setStencilReferenceValue:128]
2245 [setVertexBuffer:0x1701babe0 offset:0 atIndex:0]
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6…
2247 [popDebugGroup]
Sprites
2259 [endEncoding]
2260 0x170051a90 <- [renderCommandEncoderWithFr…
Final
2270 [addScheduledPresent:0x17022bc80]
2271 [commit]
Material
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6 instanceCount:1]
Color0
Color1
Color2
Color3
State Inspector
Bound GPU Objects
All
No Selection
FS Texture "Stencil"
FS Texture "Depth"
FS Texture "GBuffer3"
FS Texture "GBuffer2"
FS Texture "GBuffer1"
FS Texture "GBuffer0"
FS0 Texture 0x15ee1e5d0
FS1 Texture 0x15ee1c760
FS2 Texture "Shadow"
VS0 Buffer 0x1701babe0
VS1 Buffer 0x1701bbba0
VS2 Buffer 0x1701bb9e0
FS0 Buffer 0x1701bbd60
FS1 Buffer 0x1701bb120
FS2 Buffer 0x1701bb200
materialVertex (VS)
materialFrag (FS)
2246 [drawPrimitives:3 vertexStart:0 vertexCount:6 instanceCount:1]
"Material Render" (0x17809e410) materialVertex/materialFrag
Fragment Texture 0 0x15ee1e5d0
Fragment Texture 1 0x15ee1c760
Fragment Texture 2 "Shadow" (0x15ed0f760)
Fragment Sampler 0 0x1700cfb90
Fragment Buffer 0 0x1701bbd60
Fragment Buffer 1 0x1701bb120
Fragment Buffer 2 0x1701bb200
Vertex Buffer 0 0x1701babe0
Auto
Buffers 32 Buffers
Command Buffers 1 Command Buffer
Command Queues 1 Command Queue
Depth Stencil States 7 Depth Stencil States
Framebuffers 4 Framebuffers
Libraries 9 Libraries
Render Command Encoders 3 Render Command Encoders
"ShadowBuffer" (0x170051250)
"GBuffer" (0x1700529f0)
All MTL Objects

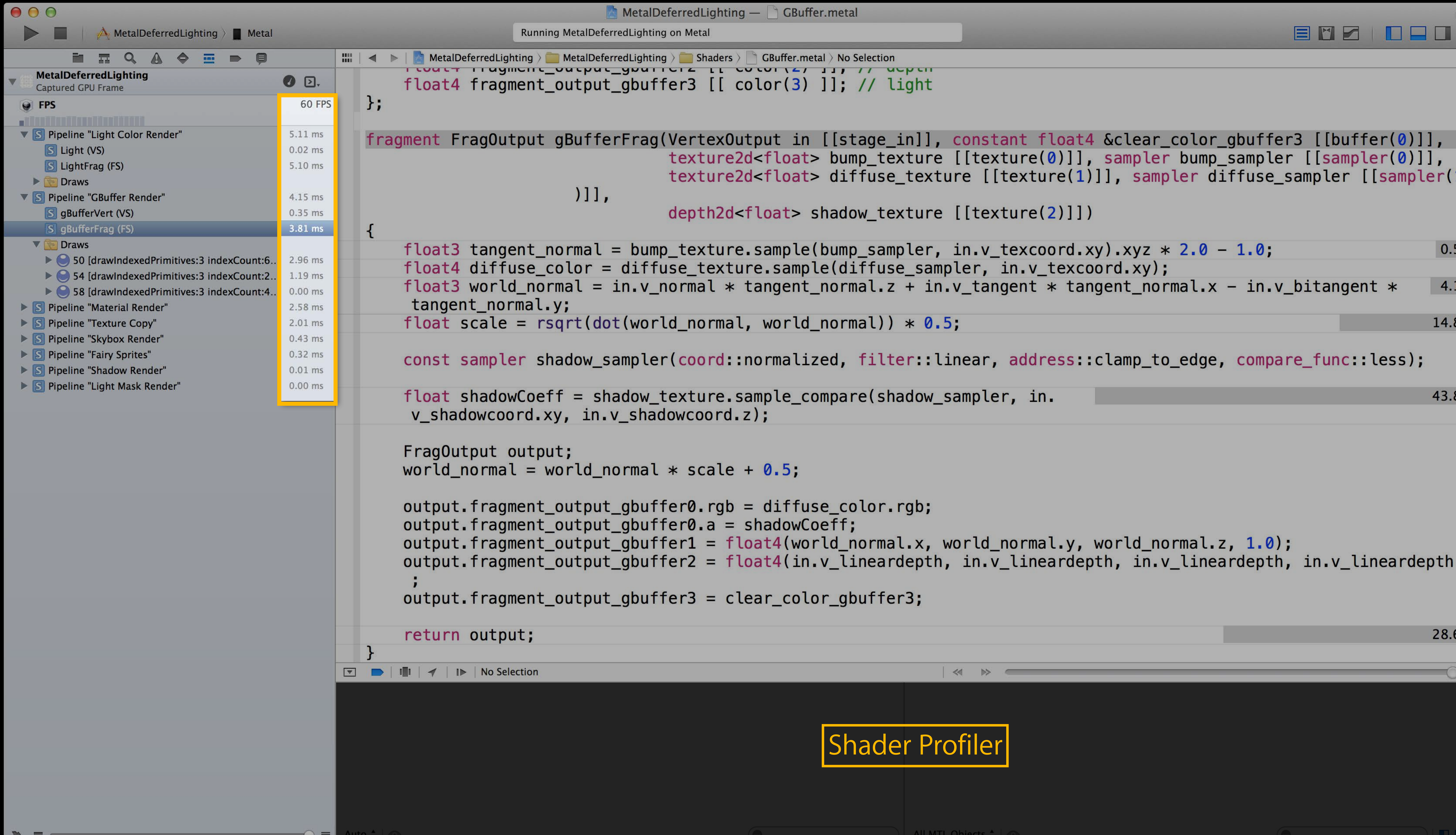

MetalDeferredLighting
Running MetalDeferredLighting on Metal
MetalDeferredLighting 〉 Metal
Graphics Report
Manual 〉 MetalDeferredLighting 〉 MetalDeferredLighting 〉 Shaders 〉 GBuffer.metal 〉 No Selection
MetalDeferredLighting
Captured GPU Frame
FPS 60 FPS
Pipeline "Light Color Re... 5.11 ms
Light (VS) 0.02 ms
LightFrag (FS) 5.10 ms
Draws
Pipeline "GBuffer Render" 4.15 ms
gBufferVert (VS) 0.35 ms
gBufferFrag (FS) 3.81 ms
Draws
50 [drawIndexedP... 2.96 ms
54 [drawIndexedP... 1.19 ms
58 [drawIndexedP... 0.00 ms
Pipeline "Material Render" 2.58 ms
Pipeline "Texture Copy" 2.01 ms
Pipeline "Skybox Render" 0.43 ms
Pipeline "Fairy Sprites" 0.32 ms
Pipeline "Shadow Render" 0.01 ms
Pipeline "Light Mask Re... 0.00 ms
Graphics
Frames Per Second 60
30
0
60
Utilization
18 % TILER
83 % RENDERER
88 % DEVICE
Frame
13.5 ms CPU
Program Performance
Program | Frame % | Current ms | 1 | 2
Pipeline "Light Color Render" 35.0% 5.11
Pipeline "GBuffer Render" 28.4% 4.15
Pipeline "Material Render" 17.7% 2.58
Pipeline "Texture Copy" 13.8% 2.01
Pipeline "Skybox Render" 2.9% 0.43
Pipeline "Fairy Sprites" 2.2% 0.32
Pipeline "Shadow Render" 0.1% 0.01
Pipeline "Light Mask Render" 0.0% 0.00
Total 14.62
fragment FragOutput gBufferFrag(VertexOutput in [[stage_in]],
constant float4 &clear_color_gbuffer3 [[buffer(0)]],
texture2d<float> bump_texture
[[texture(0)]], sampler bump_sampler
[[sampler(0)]],
texture2d<float> diffuse_texture
[[texture(1)]], sampler diffuse_sampler
[[sampler(1)]],
depth2d<float> shadow_texture
[[texture(2)]])
{
float3 tangent_normal = bump_texture.sample(bump_sampler, 0.5%
in.v_texcoord.xy).xyz * 2.0 - 1.0;
float4 diffuse_color = diffuse_texture.sample(diffuse_sampler
in.v_texcoord.xy);
float3 world_normal = in.v_normal * tangent_normal.z + in. 4.1%
v_tangent * tangent_normal.x - in.v_bitangent *
tangent_normal.y;
float scale = rsqrt(dot(world_normal, world_normal)) * 14.8
0.5;
const sampler shadow_sampler(coord::normalized, filter::linea
, address::clamp_to_edge, compare_func::less);
float shadowCoeff = shadow_texture. 43.8%
sample_compare(shadow_sampler, in.v_shadowcoord.xy, in.
v_shadowcoord.z);
FragOutput output;
world_normal = world_normal * scale + 0.5;
output.fragment_output_gbuffer0.rgb = diffuse_color.rgb;
2236 [drawIndexedPrimitives:3 indexCount:60 indexType:0 indexBuffer:0x17...
Auto
All MTL Objects

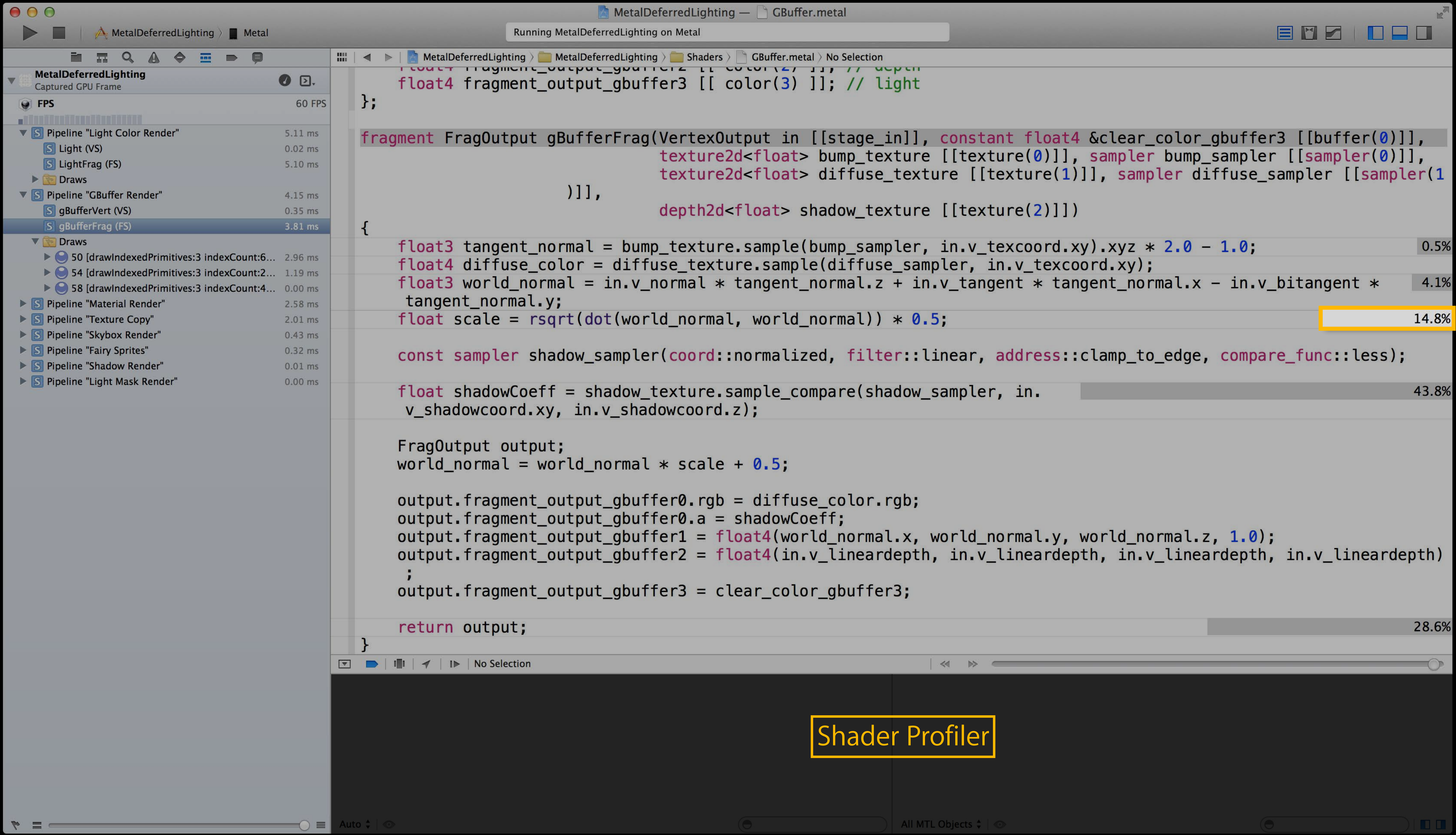


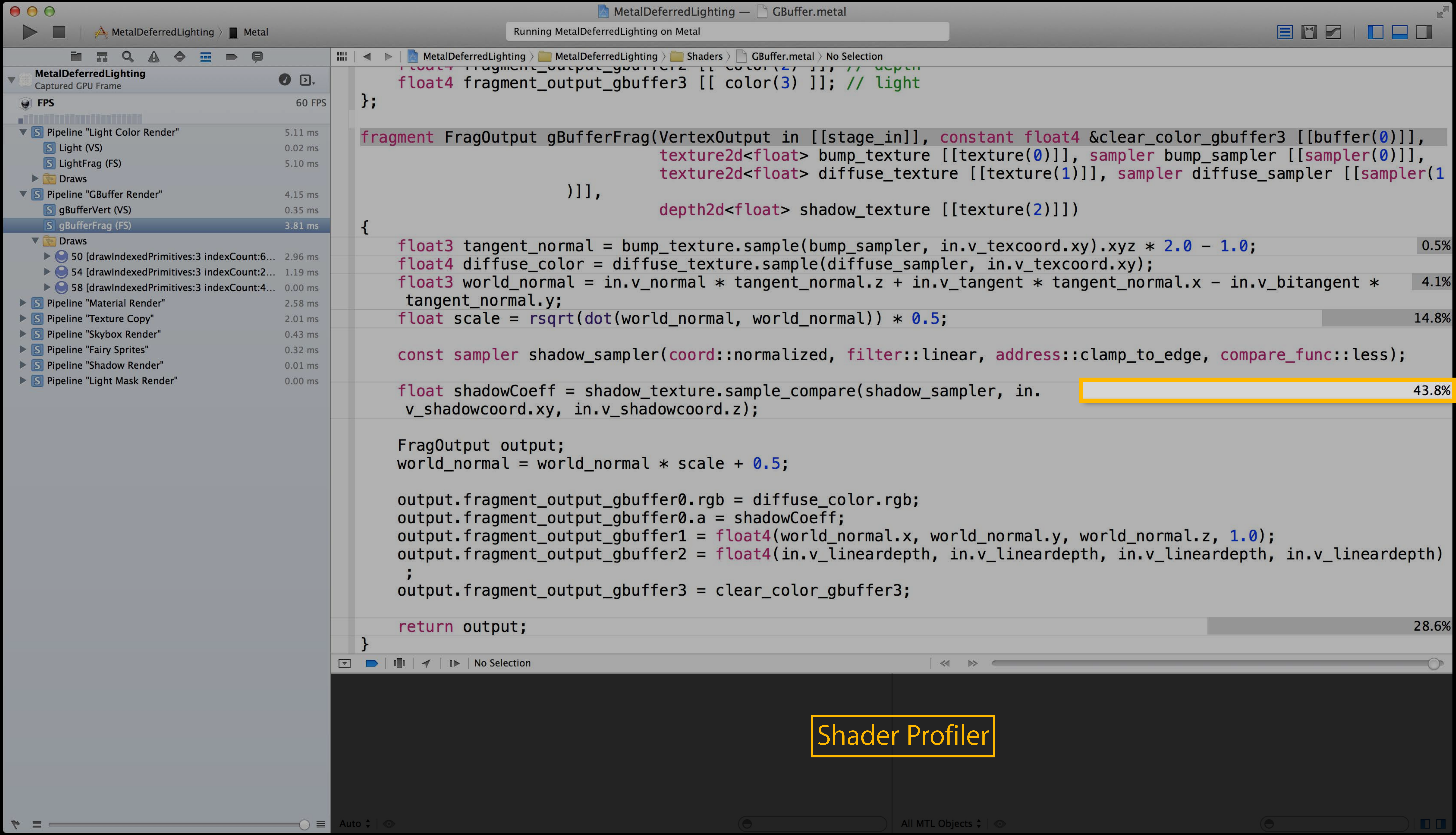
Performance Report

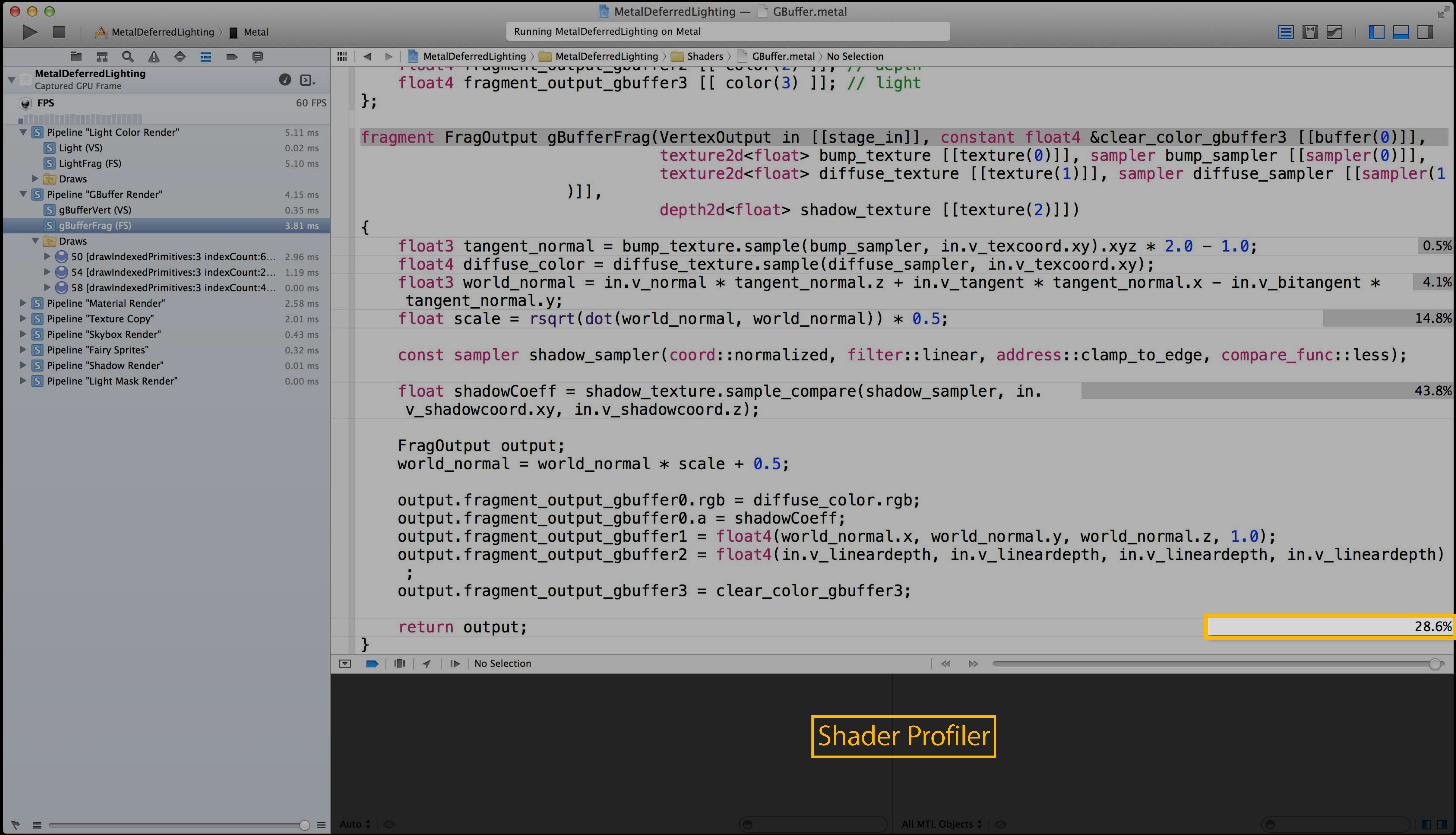

MetalDeferredLighting — GBuffer.metal
MetalDeferredLighting
Metal
Running MetalDeferredLighting on Metal
MetalDeferredLighting
MetalDeferredLighting
Shaders
GBuffer.metal
No Selection
MetalDeferredLighting
Captured GPU Frame
FPS
60 FPS
Pipeline "Light Color Render" 5.11 ms
Light (VS) 0.02 ms
LightFrag (FS) 5.10 ms
Draws
Pipeline "GBuffer Render" 4.15 ms
gBufferVert (VS) 0.35 ms
gBufferFrag (FS) 3.81 ms
Draws
50 [drawIndexedPrimitives:3 indexCount:6... 2.96 ms
54 [drawIndexedPrimitives:3 indexCount:2... 1.19 ms
58 [drawIndexedPrimitives:3 indexCount:4... 0.00 ms
Pipeline "Material Render" 2.58 ms
Pipeline "Texture Copy" 2.01 ms
Pipeline "Skybox Render" 0.43 ms
Pipeline "Fairy Sprites" 0.32 ms
Pipeline "Shadow Render" 0.01 ms
Pipeline "Light Mask Render" 0.00 ms
float4 fragment_output_gbuffer3 [[color(3)]]; // light
};
fragment FragOutput gBufferFrag(VertexOutput in [[stage_in]], constant float4 &clear_color_gbuffer3 [[buffer(0)]],
texture2d<float> bump_texture [[texture(0)]], sampler bump_sampler [[sampler(0)]],
texture2d<float> diffuse_texture [[texture(1)]], sampler diffuse_sampler [[sampler(1
)]],
depth2d<float> shadow_texture [[texture(2)]])
{
float3 tangent_normal = bump_texture.sample(bump_sampler, in.v_texcoord.xy).xyz * 2.0 - 1.0;
0.5%
float4 diffuse_color = diffuse_texture.sample(diffuse_sampler, in.v_texcoord.xy);
float3 world_normal = in.v_normal * tangent_normal.z + in.v_tangent * tangent_normal.x - in.v_bitangent *
tangent_normal.y;
4.1%
float scale = rsqrt(dot(world_normal, world_normal)) * 0.5;
14.8%
const sampler shadow_sampler(coord::normalized, filter::linear, address::clamp_to_edge, compare_func::less);
float shadowCoeff = shadow_texture.sample_compare(shadow_sampler, in.
v_shadowcoord.xy, in.v_shadowcoord.z);
43.8%
FragOutput output;
world_normal = world_normal * scale + 0.5;
output.fragment_output_gbuffer0.rgb = diffuse_color.rgb;
output.fragment_output_gbuffer0.a = shadowCoeff;
output.fragment_output_gbuffer1 = float4(world_normal.x, world_normal.y, world_normal.z, 1.0);
output.fragment_output_gbuffer2 = float4(in.v_lineardepth, in.v_lineardepth, in.v_lineardepth, in.v_lineardepth)
;
output.fragment_output_gbuffer3 = clear_color_gbuffer3;
return output;
28.6%
}
No Selection
Auto
All MTL Objects


MetalDeferredLighting — GBuffer.metal
MetalDeferredLighting 〉 Metal
Running MetalDeferredLighting on Metal
MetalDeferredLighting 〉 MetalDeferredLighting 〉 Shaders 〉 GBuffer.metal 〉 No Selection
MetalDeferredLighting
Captured GPU Frame
FPS 60 FPS
Pipeline "Light Color Render" 5.11 ms
Light (VS) 0.02 ms
LightFrag (FS) 5.10 ms
Draws
Pipeline "GBuffer Render" 4.15 ms
gBufferVert (VS) 0.35 ms
gBufferFrag (FS) 3.81 ms
Draws
50 [drawIndexedPrimitives:3 indexCount:6... 2.96 ms
54 [drawIndexedPrimitives:3 indexCount:2... 1.19 ms
58 [drawIndexedPrimitives:3 indexCount:4... 0.00 ms
Pipeline "Material Render" 2.58 ms
Pipeline "Texture Copy" 2.01 ms
Pipeline "Skybox Render" 0.43 ms
Pipeline "Fairy Sprites" 0.32 ms
Pipeline "Shadow Render" 0.01 ms
Pipeline "Light Mask Render" 0.00 ms
float4 fragment_output_gbuffer3 [[color(3)]]; // light
};
fragment FragOutput gBufferFrag(VertexOutput in [[stage_in]], constant float4 &clear_color_gbuffer3 [[buffer(0)]],
texture2d<float> bump_texture [[texture(0)]], sampler bump_sampler [[sampler(0)]],
texture2d<float> diffuse_texture [[texture(1)]], sampler diffuse_sampler [[sampler(1)]],
depth2d<float> shadow_texture [[texture(2)]])
{
float3 tangent_normal = bump_texture.sample(bump_sampler, in.v_texcoord.xy).xyz * 2.0 - 1.0; 0.5%
float4 diffuse_color = diffuse_texture.sample(diffuse_sampler, in.v_texcoord.xy);
float3 world_normal = in.v_normal * tangent_normal.z + in.v_tangent * tangent_normal.x - in.v_bitangent * tangent_normal.y; 4.1%
float scale = rsqrt(dot(world_normal, world_normal)) * 0.5; 14.8%
const sampler shadow_sampler(coord::normalized, filter::linear, address::clamp_to_edge, compare_func::less);
float shadowCoeff = shadow_texture.sample_compare(shadow_sampler, in.v_shadowcoord.xy, in.v_shadowcoord.z); 43.8%
FragOutput output;
world_normal = world_normal * scale + 0.5;
output.fragment_output_gbuffer0.rgb = diffuse_color.rgb;
output.fragment_output_gbuffer0.a = shadowCoeff;
output.fragment_output_gbuffer1 = float4(world_normal.x, world_normal.y, world_normal.z, 1.0);
output.fragment_output_gbuffer2 = float4(in.v_lineardepth, in.v_lineardepth, in.v_lineardepth, in.v_lineardepth);
output.fragment_output_gbuffer3 = clear_color_gbuffer3;
return output; 28.6%
}
No Selection
Shader Profiler
Auto
All MTL Objects


MetalDeferredLighting — GBuffer.metal
MetalDeferredLighting › Metal
Running MetalDeferredLighting on Metal
MetalDeferredLighting › MetalDeferredLighting › Shaders › GBuffer.metal › No Selection
MetalDeferredLighting
Captured GPU Frame
FPS 60 FPS
Pipeline "Light Color Render" 5.11 ms
Light (VS) 0.02 ms
LightFrag (FS) 5.10 ms
Draws
Pipeline "GBuffer Render" 4.15 ms
gBufferVert (VS) 0.35 ms
gBufferFrag (FS) 3.81 ms
Draws
50 [drawIndexedPrimitives:3 indexCount:6.. 2.96 ms
54 [drawIndexedPrimitives:3 indexCount:2.. 1.19 ms
58 [drawIndexedPrimitives:3 indexCount:4.. 0.00 ms
Pipeline "Material Render" 2.58 ms
Pipeline "Texture Copy" 2.01 ms
Pipeline "Skybox Render" 0.43 ms
Pipeline "Fairy Sprites" 0.32 ms
Pipeline "Shadow Render" 0.01 ms
Pipeline "Light Mask Render" 0.00 ms
float4 fragment_output_gbuffer3 [[color(3)]]; // light
};
fragment FragOutput gBufferFrag(VertexOutput in [[stage_in]], constant float4 &clear_color_gbuffer3 [[buffer(0)]],
texture2d<float> bump_texture [[texture(0)]], sampler bump_sampler [[sampler(0)]],
texture2d<float> diffuse_texture [[texture(1)]], sampler diffuse_sampler [[sampler(1
)]],
depth2d<float> shadow_texture [[texture(2)]])
{
float3 tangent_normal = bump_texture.sample(bump_sampler, in.v_texcoord.xy).xyz * 2.0 - 1.0; 0.5%
float4 diffuse_color = diffuse_texture.sample(diffuse_sampler, in.v_texcoord.xy);
float3 world_normal = in.v_normal * tangent_normal.z + in.v_tangent * tangent_normal.x - in.v_bitangent * tangent_normal.y; 4.1%
float scale = rsqrt(dot(world_normal, world_normal)) * 0.5; 14.8%
const sampler shadow_sampler(coord::normalized, filter::linear, address::clamp_to_edge, compare_func::less);
float shadowCoeff = shadow_texture.sample_compare(shadow_sampler, in.v_shadowcoord.xy, in.v_shadowcoord.z); 43.8%
FragOutput output;
world_normal = world_normal * scale + 0.5;
output.fragment_output_gbuffer0.rgb = diffuse_color.rgb;
output.fragment_output_gbuffer0.a = shadowCoeff;
output.fragment_output_gbuffer1 = float4(world_normal.x, world_normal.y, world_normal.z, 1.0);
output.fragment_output_gbuffer2 = float4(in.v_lineardepth, in.v_lineardepth, in.v_lineardepth, in.v_lineardepth);
output.fragment_output_gbuffer3 = clear_color_gbuffer3;
return output; 28.6%
}
No Selection
Shader Profiler
Auto
All MTL Objects


MetalDeferredLighting — GBuffer.metal
MetalDeferredLighting
Metal
Running MetalDeferredLighting on Metal
MetalDeferredLighting
MetalDeferredLighting
Shaders
GBuffer.metal
No Selection
MetalDeferredLighting
Captured GPU Frame
FPS
60 FPS
Pipeline "Light Color Render" 5.11 ms
Light (VS) 0.02 ms
LightFrag (FS) 5.10 ms
Draws
Pipeline "GBuffer Render" 4.15 ms
gBufferVert (VS) 0.35 ms
gBufferFrag (FS) 3.81 ms
Draws
50 [drawIndexedPrimitives:3 indexCount:6... 2.96 ms
54 [drawIndexedPrimitives:3 indexCount:2... 1.19 ms
58 [drawIndexedPrimitives:3 indexCount:4... 0.00 ms
Pipeline "Material Render" 2.58 ms
Pipeline "Texture Copy" 2.01 ms
Pipeline "Skybox Render" 0.43 ms
Pipeline "Fairy Sprites" 0.32 ms
Pipeline "Shadow Render" 0.01 ms
Pipeline "Light Mask Render" 0.00 ms
float4 fragment_output_gbuffer3 [[color(3)]]; // light
};
fragment FragOutput gBufferFrag(VertexOutput in [[stage_in]], constant float4 &clear_color_gbuffer3 [[buffer(0)]],
texture2d<float> bump_texture [[texture(0)]], sampler bump_sampler [[sampler(0)]],
texture2d<float> diffuse_texture [[texture(1)]], sampler diffuse_sampler [[sampler(1
)]],
depth2d<float> shadow_texture [[texture(2)]])
{
float3 tangent_normal = bump_texture.sample(bump_sampler, in.v_texcoord.xy).xyz * 2.0 - 1.0; 0.5%
float4 diffuse_color = diffuse_texture.sample(diffuse_sampler, in.v_texcoord.xy);
float3 world_normal = in.v_normal * tangent_normal.z + in.v_tangent * tangent_normal.x - in.v_bitangent * 4.1%
tangent_normal.y;
float scale = rsqrt(dot(world_normal, world_normal)) * 0.5; 14.8%
const sampler shadow_sampler(coord::normalized, filter::linear, address::clamp_to_edge, compare_func::less);
float shadowCoeff = shadow_texture.sample_compare(shadow_sampler, in. 43.8%
v_shadowcoord.xy, in.v_shadowcoord.z);
FragOutput output;
world_normal = world_normal * scale + 0.5;
output.fragment_output_gbuffer0.rgb = diffuse_color.rgb;
output.fragment_output_gbuffer0.a = shadowCoeff;
output.fragment_output_gbuffer1 = float4(world_normal.x, world_normal.y, world_normal.z, 1.0);
output.fragment_output_gbuffer2 = float4(in.v_lineardepth, in.v_lineardepth, in.v_lineardepth, in.v_lineardepth)
;
output.fragment_output_gbuffer3 = clear_color_gbuffer3;
return output; 28.6%
}
No Selection
Shader Profiler
Auto
All MTL Objects


MetalDeferredLighting — GBuffer.metal
MetalDeferredLighting › Metal
Running MetalDeferredLighting on Metal
MetalDeferredLighting › MetalDeferredLighting › Shaders › GBuffer.metal › No Selection
MetalDeferredLighting
Captured GPU Frame
FPS 60 FPS
Pipeline "Light Color Render" 5.11 ms
Light (VS) 0.02 ms
LightFrag (FS) 5.10 ms
Draws
Pipeline "GBuffer Render" 4.15 ms
gBufferVert (VS) 0.35 ms
gBufferFrag (FS) 3.81 ms
Draws
50 [drawIndexedPrimitives:3 indexCount:6... 2.96 ms
54 [drawIndexedPrimitives:3 indexCount:2... 1.19 ms
58 [drawIndexedPrimitives:3 indexCount:4... 0.00 ms
Pipeline "Material Render" 2.58 ms
Pipeline "Texture Copy" 2.01 ms
Pipeline "Skybox Render" 0.43 ms
Pipeline "Fairy Sprites" 0.32 ms
Pipeline "Shadow Render" 0.01 ms
Pipeline "Light Mask Render" 0.00 ms
float4 fragment_output_gbuffer3 [[color(3)]]; // light
};
fragment FragOutput gBufferFrag(VertexOutput in [[stage_in]], constant float4 &clear_color_gbuffer3 [[buffer(0)]],
texture2d<float> bump_texture [[texture(0)]], sampler bump_sampler [[sampler(0)]],
texture2d<float> diffuse_texture [[texture(1)]], sampler diffuse_sampler [[sampler(1
)]],
depth2d<float> shadow_texture [[texture(2)]])
{
float3 tangent_normal = bump_texture.sample(bump_sampler, in.v_texcoord.xy).xyz * 2.0 - 1.0; 0.5%
float4 diffuse_color = diffuse_texture.sample(diffuse_sampler, in.v_texcoord.xy);
float3 world_normal = in.v_normal * tangent_normal.z + in.v_tangent * tangent_normal.x - in.v_bitangent * tangent_normal.y; 4.1%
float scale = rsqrt(dot(world_normal, world_normal)) * 0.5; 14.8%
const sampler shadow_sampler(coord::normalized, filter::linear, address::clamp_to_edge, compare_func::less);
float shadowCoeff = shadow_texture.sample_compare(shadow_sampler, in.v_shadowcoord.xy, in.v_shadowcoord.z); 43.8%
FragOutput output;
world_normal = world_normal * scale + 0.5;
output.fragment_output_gbuffer0.rgb = diffuse_color.rgb;
output.fragment_output_gbuffer0.a = shadowCoeff;
output.fragment_output_gbuffer1 = float4(world_normal.x, world_normal.y, world_normal.z, 1.0);
output.fragment_output_gbuffer2 = float4(in.v_lineardepth, in.v_lineardepth, in.v_lineardepth, in.v_lineardepth);
output.fragment_output_gbuffer3 = clear_color_gbuffer3;
return output; 28.6%
}
No Selection
Shader Profiler
Auto
All MTL Objects


MetalDeferredLighting — GBuffer.metal
Running MetalDeferredLighting on Metal
MetalDeferredLighting
Captured GPU Frame
FPS 60 FPS
Pipeline "Light Color Render" 5.11 ms
Light (VS) 0.02 ms
LightFrag (FS) 5.10 ms
Draws
Pipeline "GBuffer Render" 4.15 ms
gBufferVert (VS) 0.35 ms
gBufferFrag (FS) 3.81 ms
Draws
50 [drawIndexedPrimitives:3 indexCount:6... 2.96 ms
54 [drawIndexedPrimitives:3 indexCount:2... 1.19 ms
58 [drawIndexedPrimitives:3 indexCount:4... 0.00 ms
Pipeline "Material Render" 2.58 ms
Pipeline "Texture Copy" 2.01 ms
Pipeline "Skybox Render" 0.43 ms
Pipeline "Fairy Sprites" 0.32 ms
Pipeline "Shadow Render" 0.01 ms
Pipeline "Light Mask Render" 0.00 ms
MetalDeferredLighting › MetalDeferredLighting › Shaders › GBuffer.metal › No Selection
float4 fragment_output_gbuffer3 [[color(3)]]; // light
};
fragment FragOutput gBufferFrag(VertexOutput in [[stage_in]], constant float4 &clear_color_gbuffer3 [[buffer(0)]],
texture2d<float> bump_texture [[texture(0)]], sampler bump_sampler [[sampler(0)]],
texture2d<float> diffuse_texture [[texture(1)]], sampler diffuse_sampler [[sampler(1)]],
depth2d<float> shadow_texture [[texture(2)]])
{
float3 tangent_normal = bump_texture.sample(bump_sampler, in.v_texcoord.xy).xyz * 2.0 - 1.0; 0.5%
float4 diffuse_color = diffuse_texture.sample(diffuse_sampler, in.v_texcoord.xy);
float3 world_normal = in.v_normal * tangent_normal.z + in.v_tangent * tangent_normal.x - in.v_bitangent * tangent_normal.y; 4.1%
float scale = rsqrt(dot(world_normal, world_normal)) * 0.5; 14.8%
const sampler shadow_sampler(coord::normalized, filter::linear, address::clamp_to_edge, compare_func::less);
float shadowCoeff = shadow_texture.sample_compare(shadow_sampler, in.v_shadowcoord.xy, in.v_shadowcoord.z); 43.8%
FragOutput output;
world_normal = world_normal * scale + 0.5;
output.fragment_output_gbuffer0.rgb = diffuse_color.rgb;
output.fragment_output_gbuffer0.a = shadowCoeff;
output.fragment_output_gbuffer1 = float4(world_normal.x, world_normal.y, world_normal.z, 1.0);
output.fragment_output_gbuffer2 = float4(in.v_lineardepth, in.v_lineardepth, in.v_lineardepth, in.v_lineardepth);
output.fragment_output_gbuffer3 = clear_color_gbuffer3;
return output; 28.6%
}
No Selection
Shader Profiler
Auto
All MTL Objects

MetalDeferredLighting — GBuffer.metal

MetalDeferredLighting 〉 Metal | MetalDeferredLighting | Build **Failed** | Today at 11:28 AM ⚠1 ❗1

gBufferFrag(VertexOutput in [[stage_in]], constant float4 &clear_color_gbuffer3 [[buffer(0)]], texture2d<float> bump_texture [[texture(0)]], sampler bump_sampler [[sampler(0)]], texture2d<float> diffus…

By File | By Type

MetalDeferredLighting
2 issues

GBuffer.metal

Semantic Issue
Cannot initialize a variable of type 'float4' (aka 'vector_float4') with an rvalue of type 'floa…

Value Conversion Issue
Implicit conversion from 'double' to 'char' changes value from 4.1 to 4

```
struct FragOutput
{
    float4 fragment_output_gbuffer0 [[ color(0) ]]; // albedo/final
    float4 fragment_output_gbuffer1 [[ color(1) ]]; // normal
    float4 fragment_output_gbuffer2 [[ color(2) ]]; // depth
    float4 fragment_output_gbuffer3 [[ color(3) ]]; // light
};

fragment FragOutput gBufferFrag(VertexOutput in [[stage_in]], constant float4 &clear_color_gbuffer3 [[buffer(0)]],
                                texture2d<float> bump_texture [[texture(0)]], sampler bump_sampler [[sampler(0)]],
                                texture2d<float> diffuse_texture [[texture(1)]], sampler diffuse_sampler [[sampler(1)]],
                                depth2d<float> shadow_texture [[texture(2)]])
{
    float3 tangent_normal = bump_texture.sample(bump_sampler, in.v_texcoord.xy).xyz * 2.0 - 1.0;
    float4 diffuse_color = diffuse_texture.sample(diffuse_sampler, in.v_texcoord.xy);
    float4 world_normal = in.v_normal * tangent_normal.z + in.v_tangent * tangent_normal.x - in.v_bitangent *
     tangent_normal.y;
    float scale = rsqrt(dot(world_normal, world_normal)) * 0.5;   ❗ Cannot initialize a variable of type 'float4' (aka 'vector_float4') with an rvalue of t…

    const sampler shadow_sampler(coord::normalized, filter::linear, address::clamp_to_edge, compare_func::less);

    const char kSampleXCount = 4.1;   ⚠ Implicit conversion from 'double' to 'char' changes value from 4.1 to 4
    const char kSampleYCount = 4;
    float sum = 0;

    for (char dy = -kSampleYCount/2; dy < kSampleYCount/2; ++dy)
    {
        for (char dx = -kSampleXCount/2; dx < kSampleXCount/2; ++dx)
        {
            sum += shadow_texture.sample_compare(shadow_sampler, in.v_shadowcoord.xy, in.v_shadowcoord.z, char2(dx, dy));
        }
    }

    float shadowCoeff = sum / ((float)kSampleXCount * kSampleYCount);

    FragOutput output;
    world_normal = world_normal * scale + 0.5;

    output.fragment_output_gbuffer0.rgb = diffuse_color.rgb;
    output.fragment_output_gbuffer0.a = shadowCoeff;
    output.fragment_output_gbuffer1 = float4(world_normal.x, world_normal.y, world_normal.z, 1.0);
    output.fragment_output_gbuffer2 = float4(in.v_lineardepth, in.v_lineardepth, in.v_lineardepth, in.v_lineardepth);
    output.fragment_output_gbuffer3 = clear_color_gbuffer3;
```

Demo

“The Collectables” on Metal

Sean Tracey
Crytek

Summary

Summary

Low overhead, high performance GPU programming API

Summary

Low overhead, high performance GPU programming API

Up to 10x more draw calls

Summary

Low overhead, high performance GPU programming API

Up to 10x more draw calls

Designed for A7 and iOS

Summary

Low overhead, high performance GPU programming API

Up to 10x more draw calls

Designed for A7 and iOS

Streamlined for modern GPU features

Summary

Low overhead, high performance GPU programming API

Up to 10x more draw calls

Designed for A7 and iOS

Streamlined for modern GPU features

Fine-grained control

Summary

Low overhead, high performance GPU programming API

Up to 10x more draw calls

Designed for A7 and iOS

Streamlined for modern GPU features

Fine-grained control

Precompiled shaders

Summary

Low overhead, high performance GPU programming API

Up to 10x more draw calls

Designed for A7 and iOS

Streamlined for modern GPU features

Fine-grained control

Precompiled shaders

Fantastic developer tools

Summary

Low overhead, high performance GPU programming API

Up to 10x more draw calls

Designed for A7 and iOS

Streamlined for modern GPU features

Fine-grained control

Precompiled shaders

Fantastic developer tools

Enables entirely new class of games

More Information

Filip Iliescu
Graphics and Games Technologies Evangelist
filiescu@apple.com

Allan Schaffer
Graphics and Games Technologies Evangelist
aschaffer@apple.com

Documentation
http://developer.apple.com

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Working with Metal—Fundamentals	Pacific Heights	Wednesday 10:15AM
Working with Metal—Advanced	Pacific Heights	Wednesday 11:30AM

Labs

• Metal Lab	Graphics and Games Lab A	Wednesday 2:00PM
• Metal Lab	Graphics and Games Lab B	Thursday 10:15AM

WWDC14