

# What's New in Metal, Part 1

Session 603

Rav Dhiraj GPU Software

# Metal at WWDC

## What's New in Metal, Part 1

- Metal in Review
- New Features
- Metal and App Thinning

## What's New in Metal, Part 2

- Introducing MetalKit
- Metal Performance Shaders

## Metal Performance Optimization Techniques

- Metal System Trace Tool
- Metal Best Practices

# Metal at WWDC

## What's New in Metal, Part 1

- Metal in Review
- New Features
- Metal and App Thinning

## What's New in Metal, Part 2

- Introducing MetalKit
- Metal Performance Shaders

## Metal Performance Optimization Techniques

- Metal System Trace Tool
- Metal Best Practices

# Metal at WWDC

## What's New in Metal, Part 1

- Metal in Review
- New Features
- Metal and App Thinning

## What's New in Metal, Part 2

- Introducing MetalKit
- Metal Performance Shaders

## Metal Performance Optimization Techniques

- Metal System Trace Tool
- Metal Best Practices

# Metal in Review

# Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading





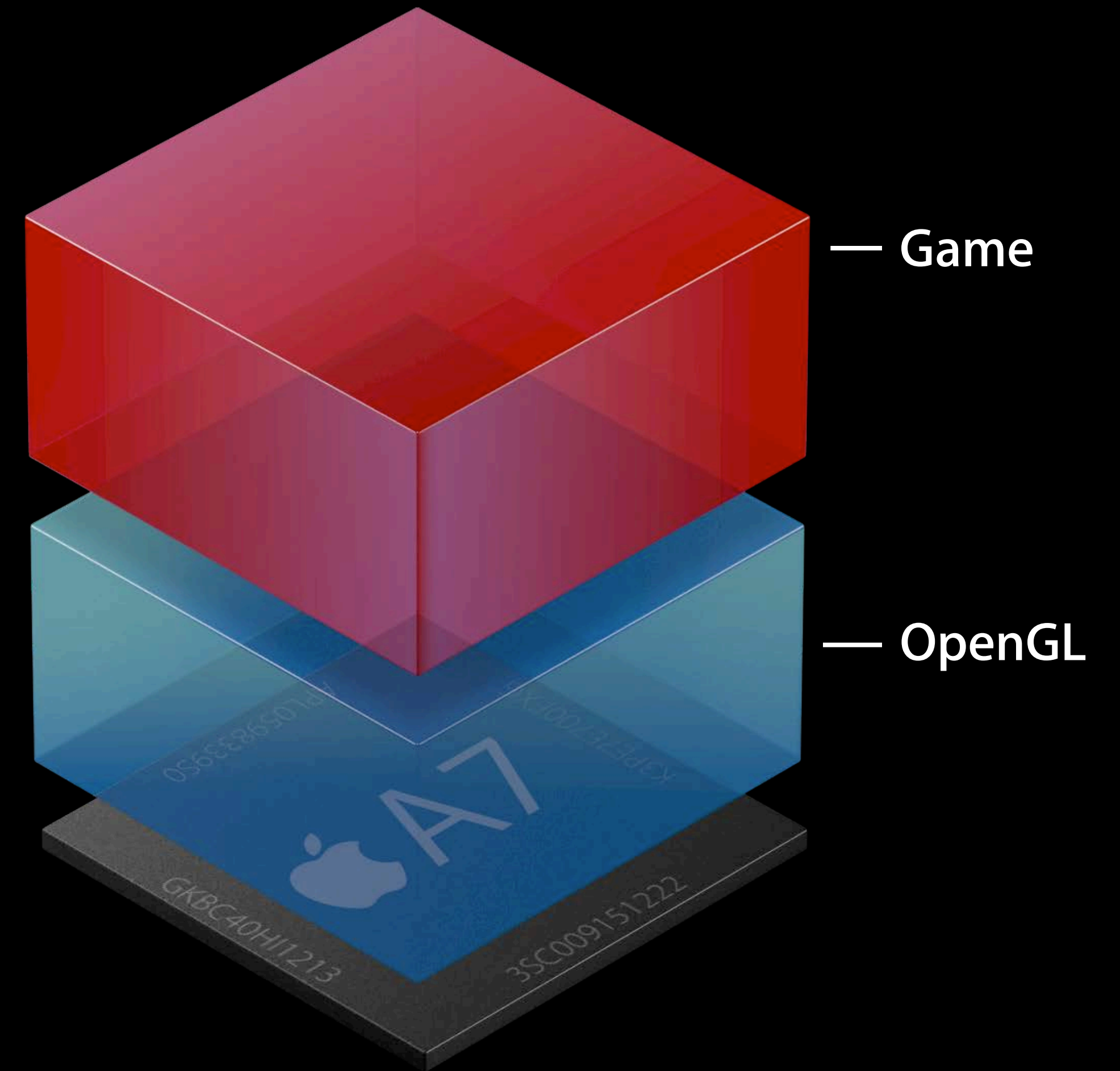
# Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading



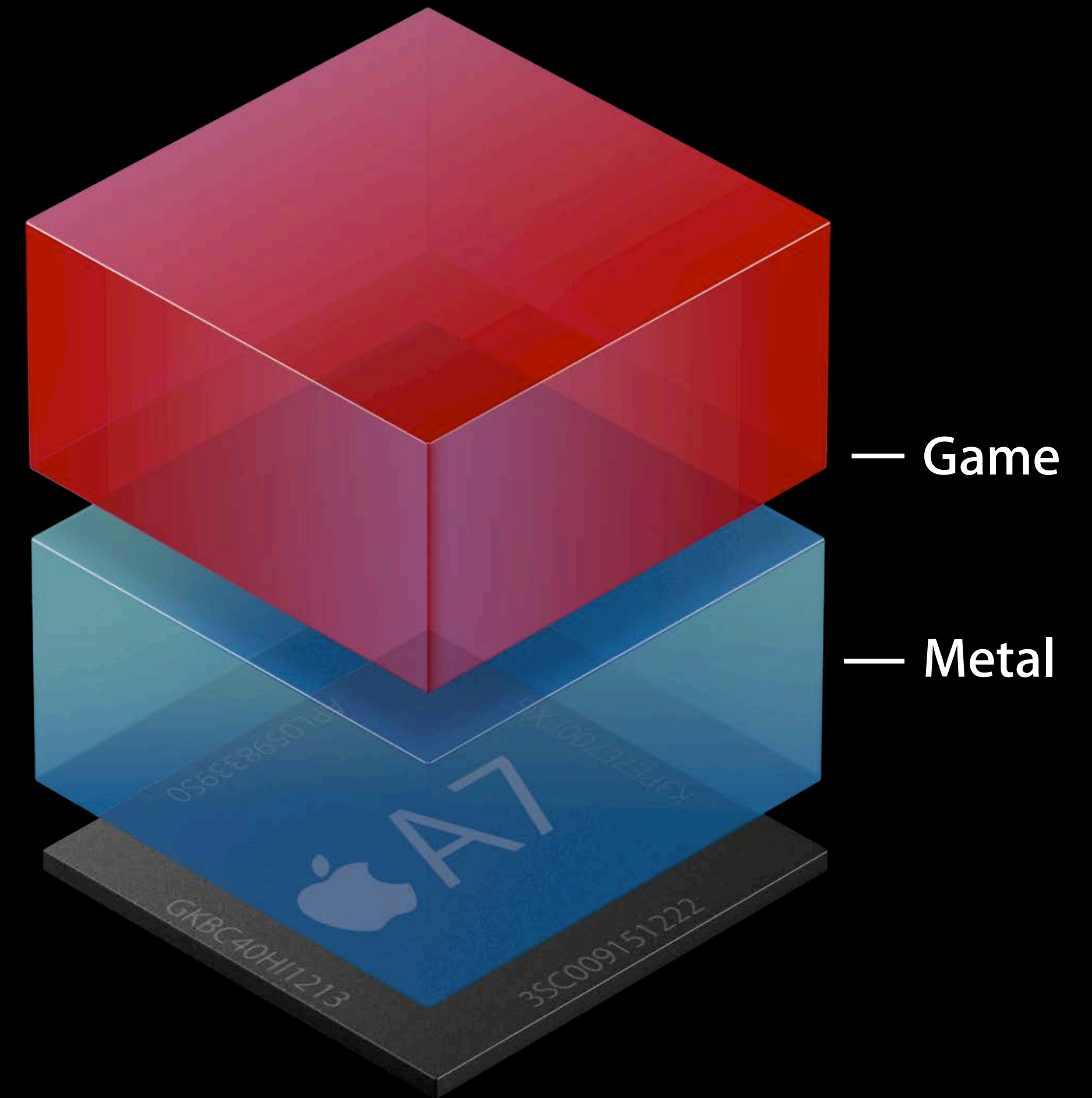
# Metal

Dramatically reduced overhead

Precompiled shaders

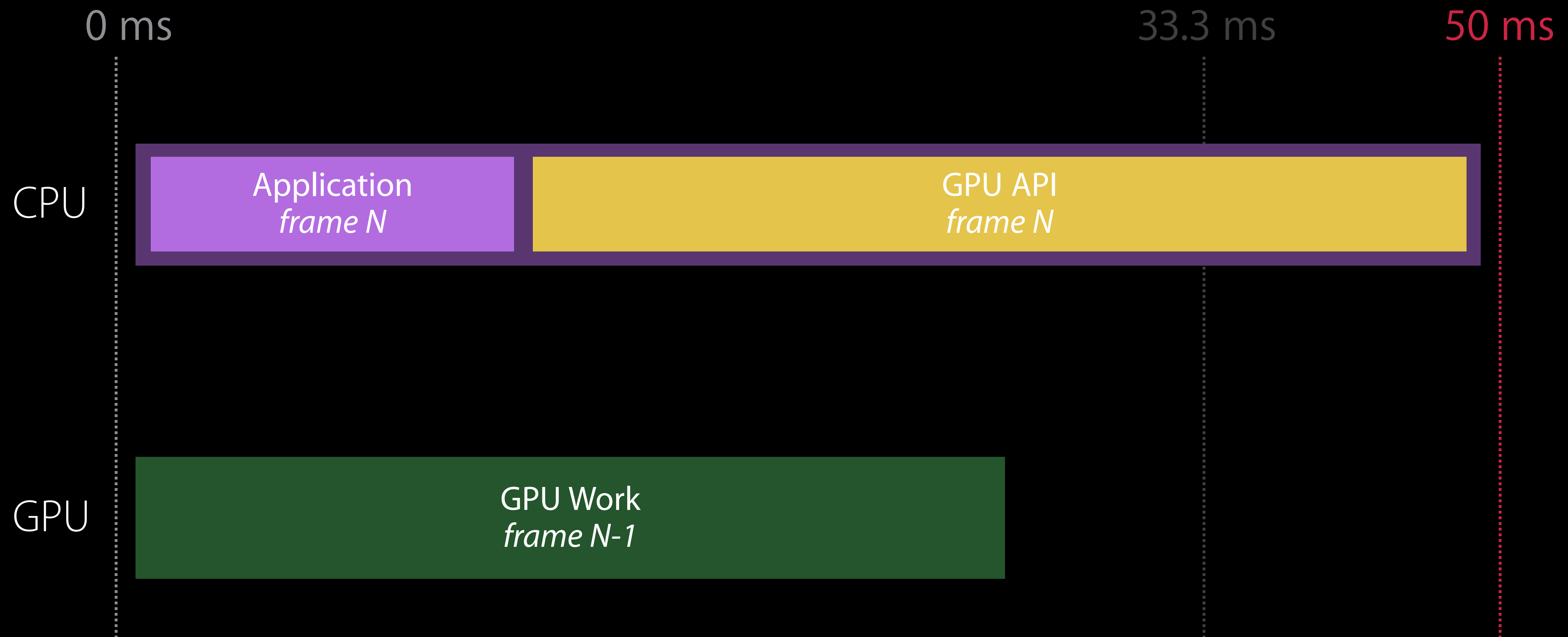
Graphics and compute

Efficient multithreading

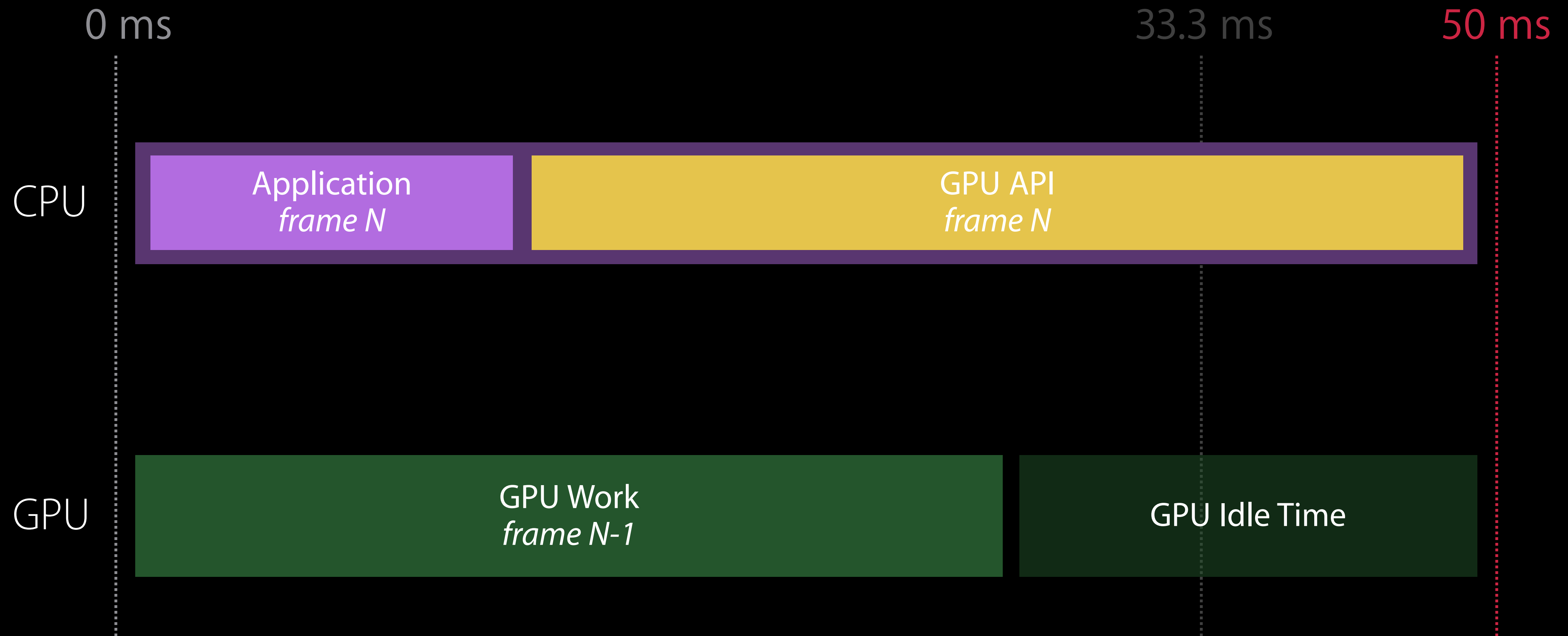




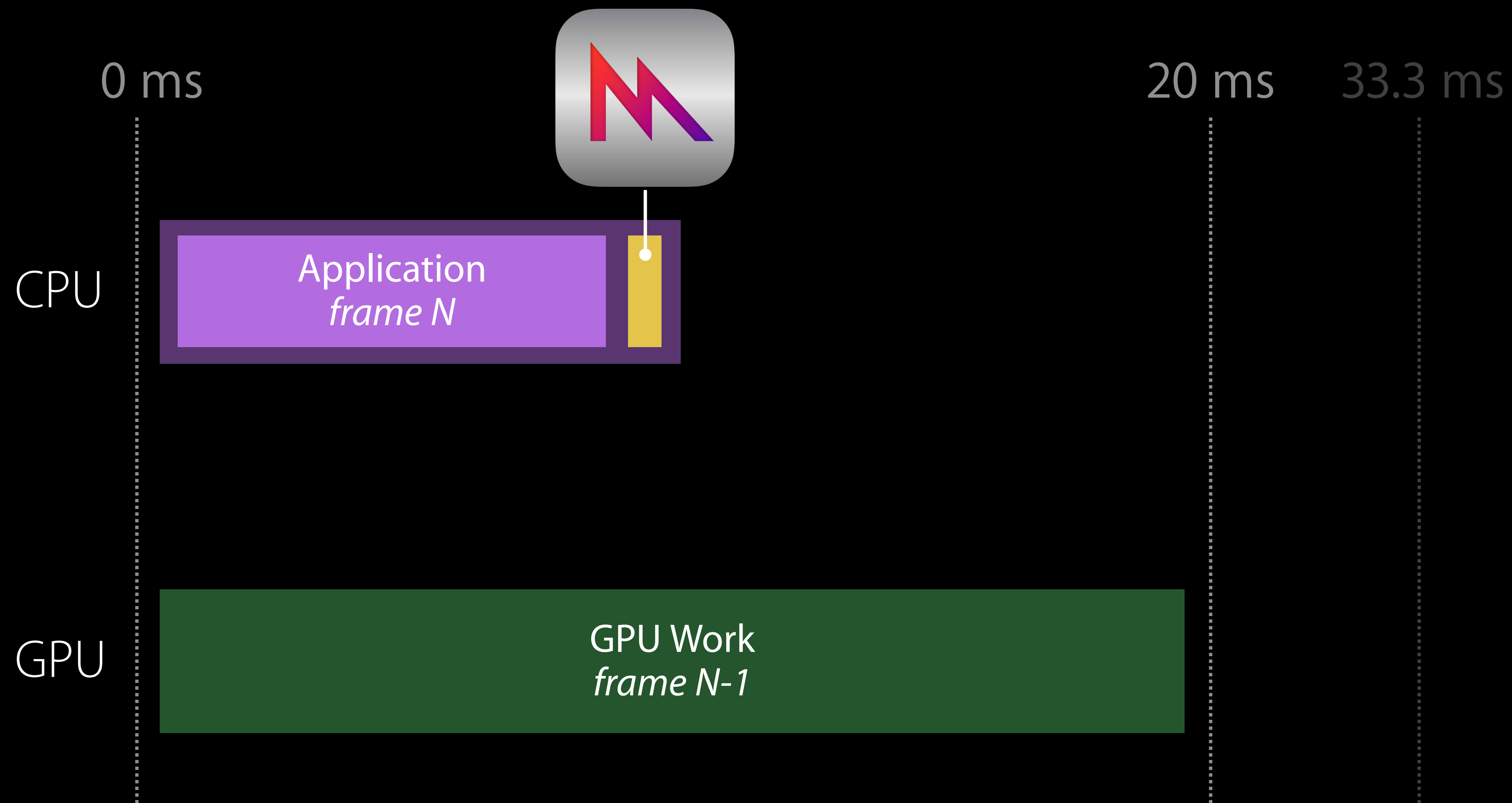
# Frame Time with CPU as the Bottleneck



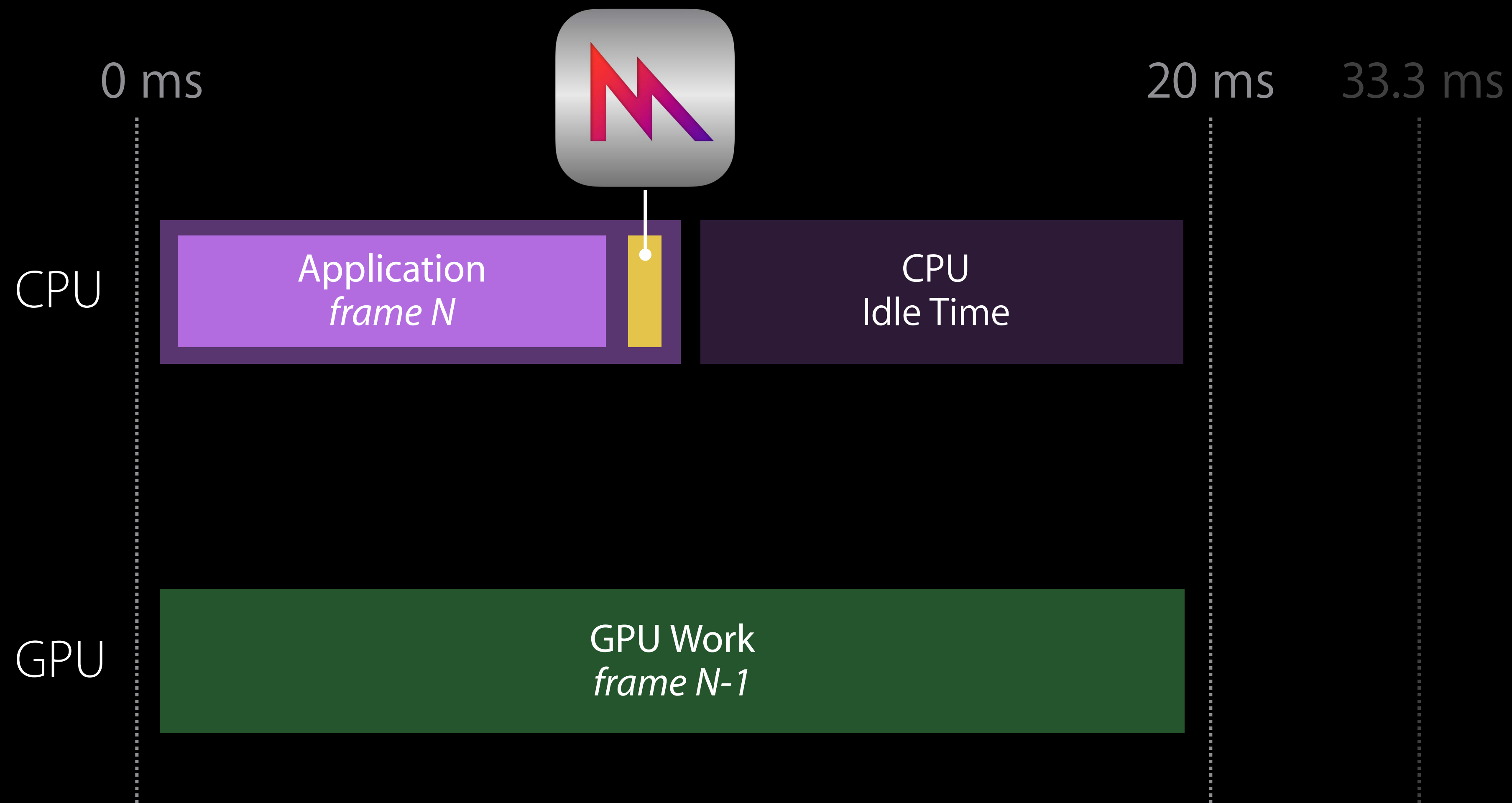
# Frame Time with CPU as the Bottleneck



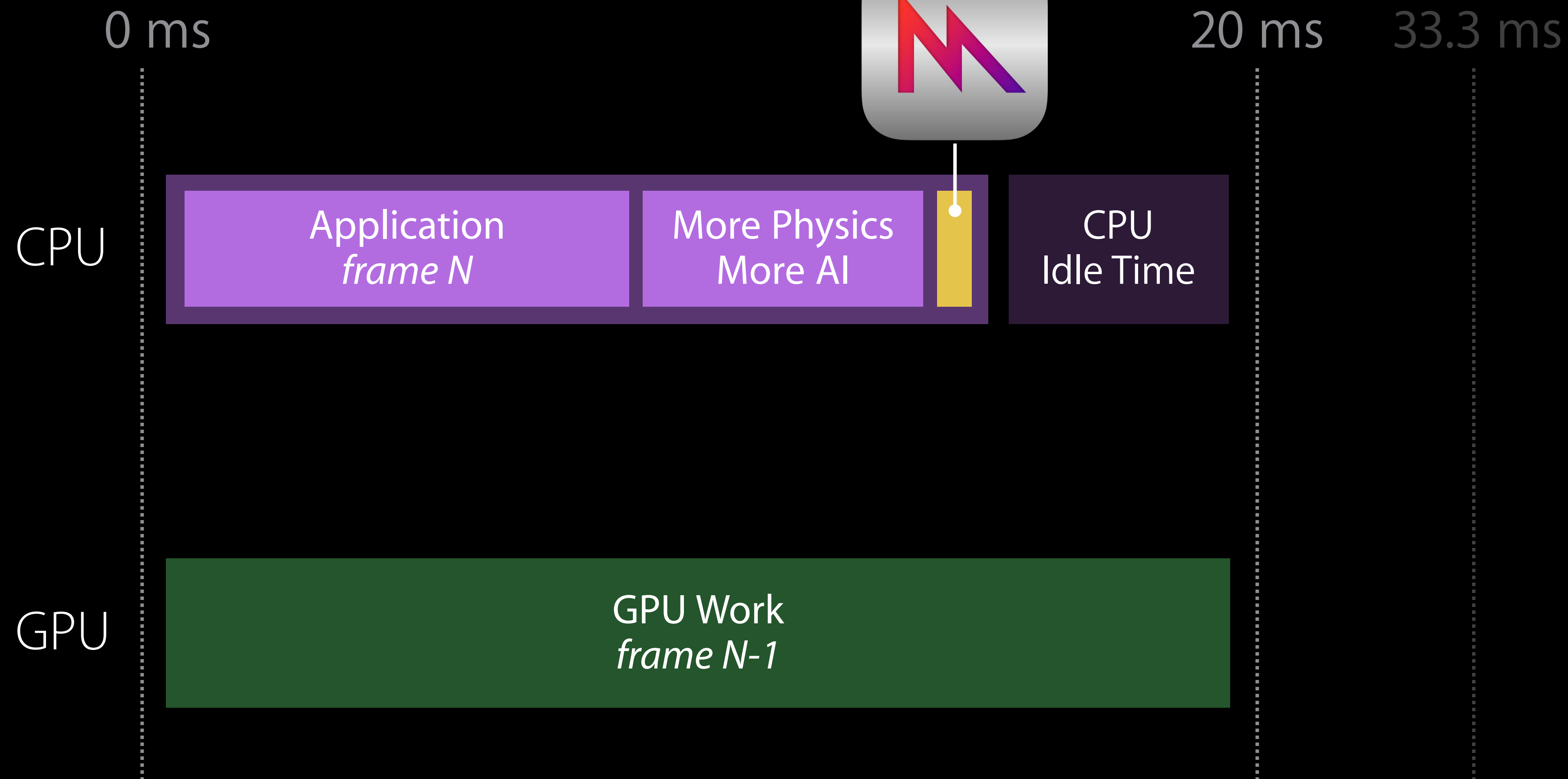
# Metal Reduces GPU API Overhead



# Metal Reduces GPU API Overhead

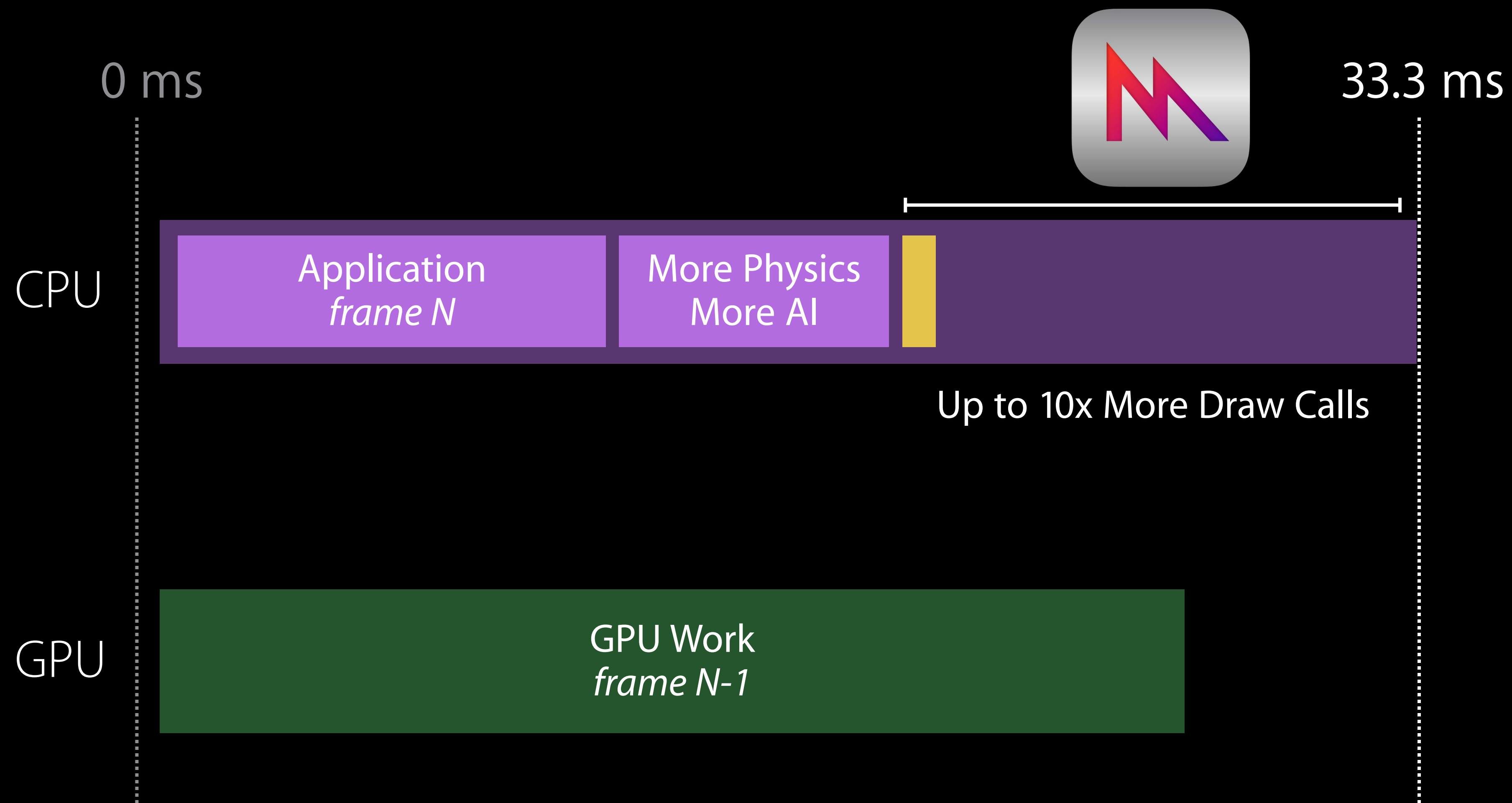


# Improve Your Game

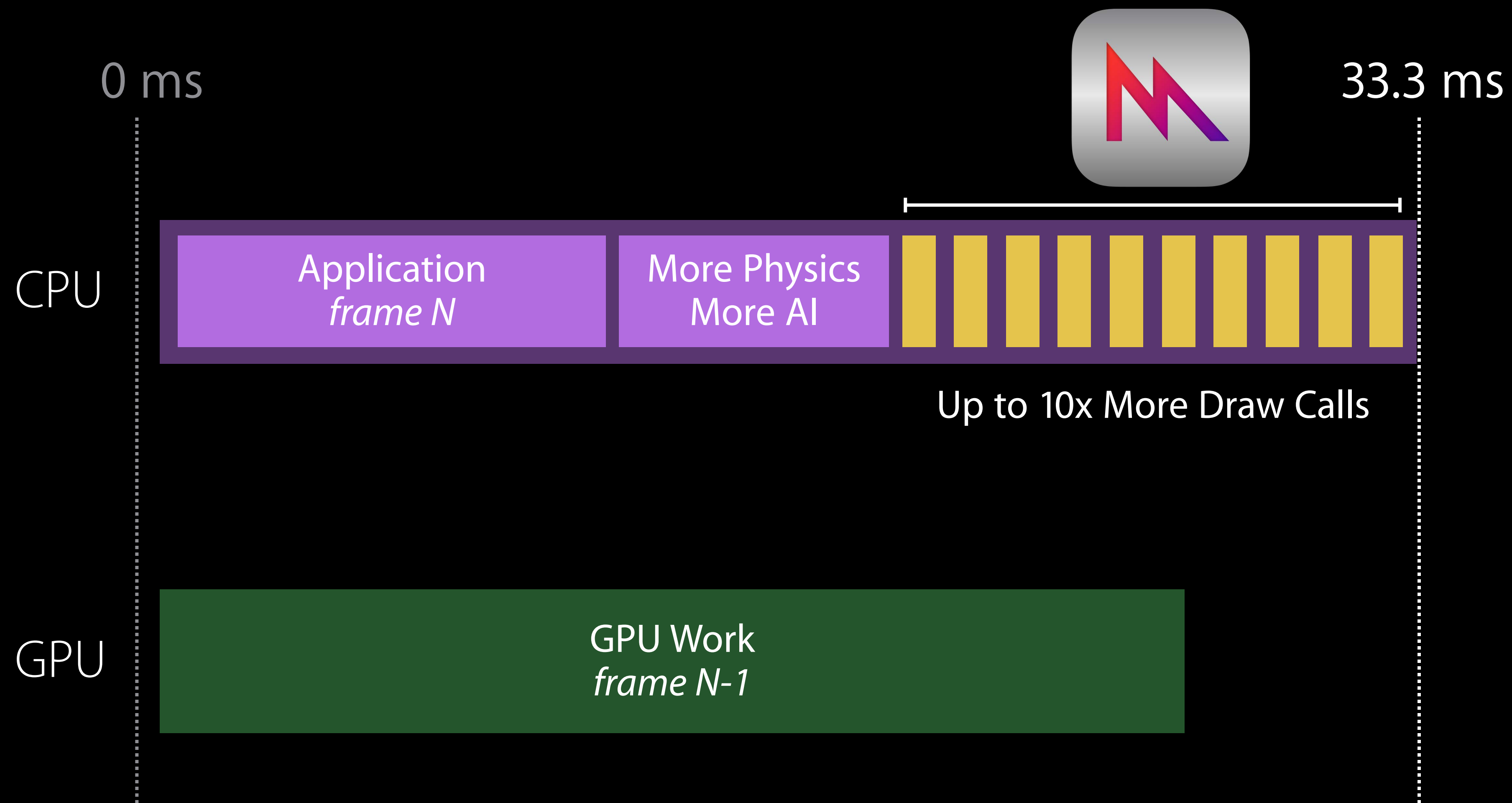




# Or Issue More Draw Calls



# Or Issue More Draw Calls



# Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading

Build Time  
"Never"

Load Time  
Infrequent

Draw Time  
1000s per  
Frame

Shader  
Compilation

State  
Validation

# Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading

Build Time  
"Never"

Load Time  
Infrequent

Draw Time  
1000s per  
Frame

Shader  
Compilation

State  
Validation

# Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading

Build Time  
"Never"

Load Time  
Infrequent

Draw Time  
1000s per  
Frame

Shader  
Compilation

State  
Validation



# Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading

Render Command  
Encoder

Compute  
Command Encoder

Render Command  
Encoder



Command Buffer

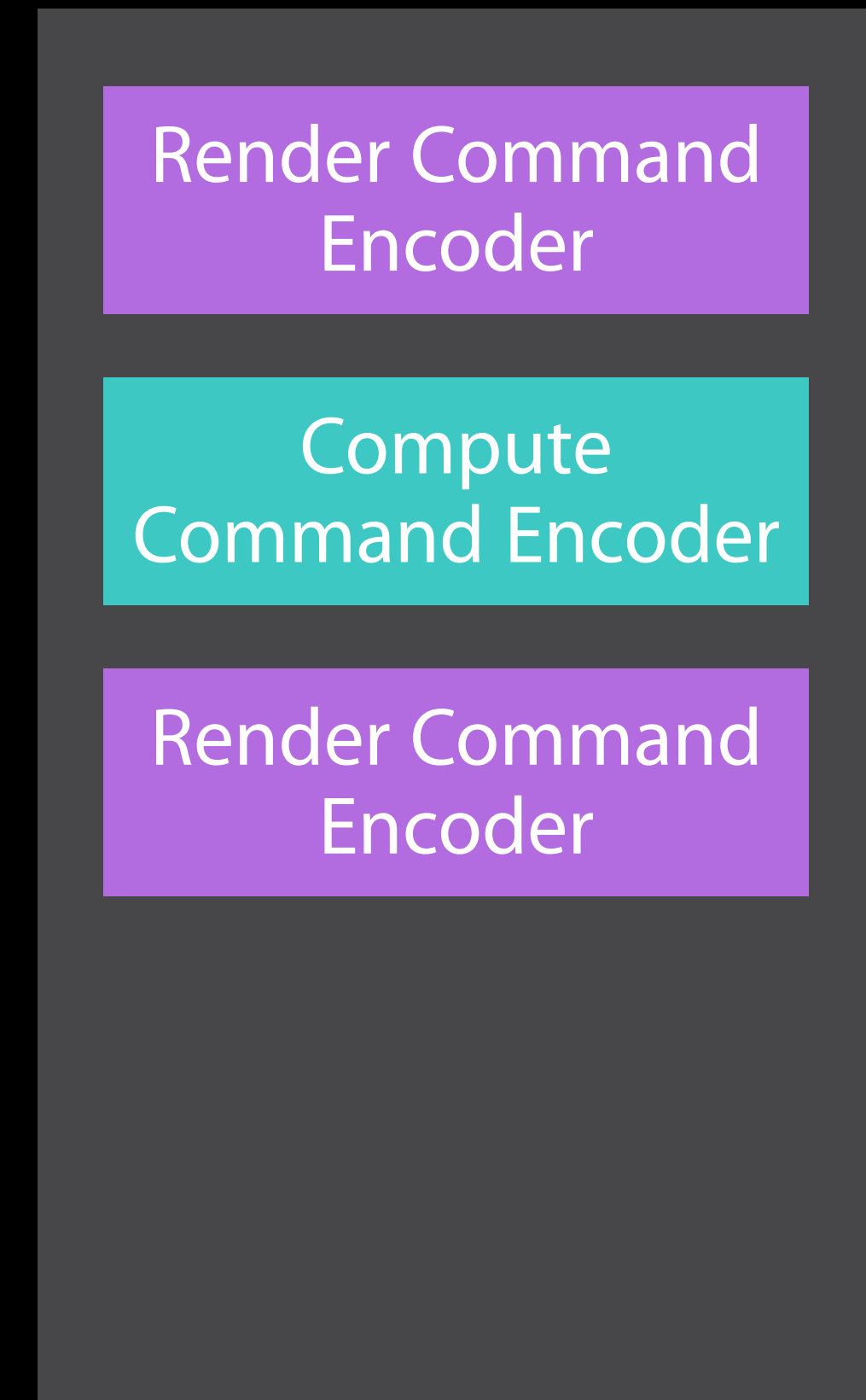
# Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading



Command Buffer

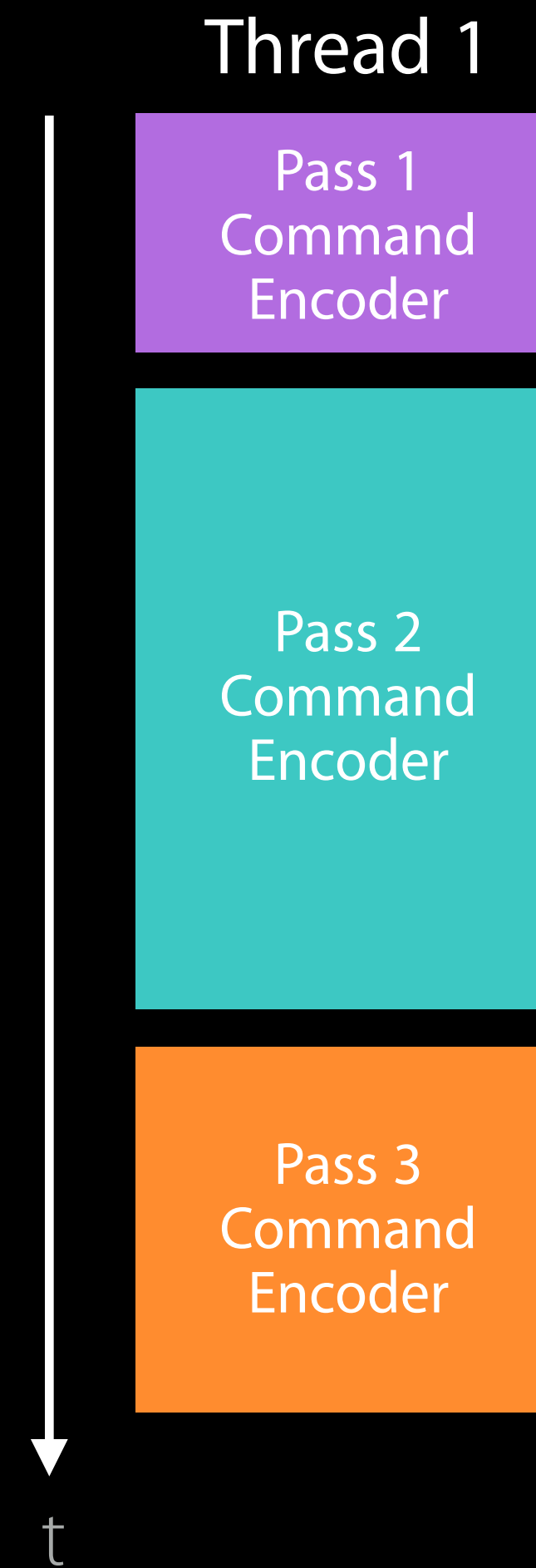
# Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading



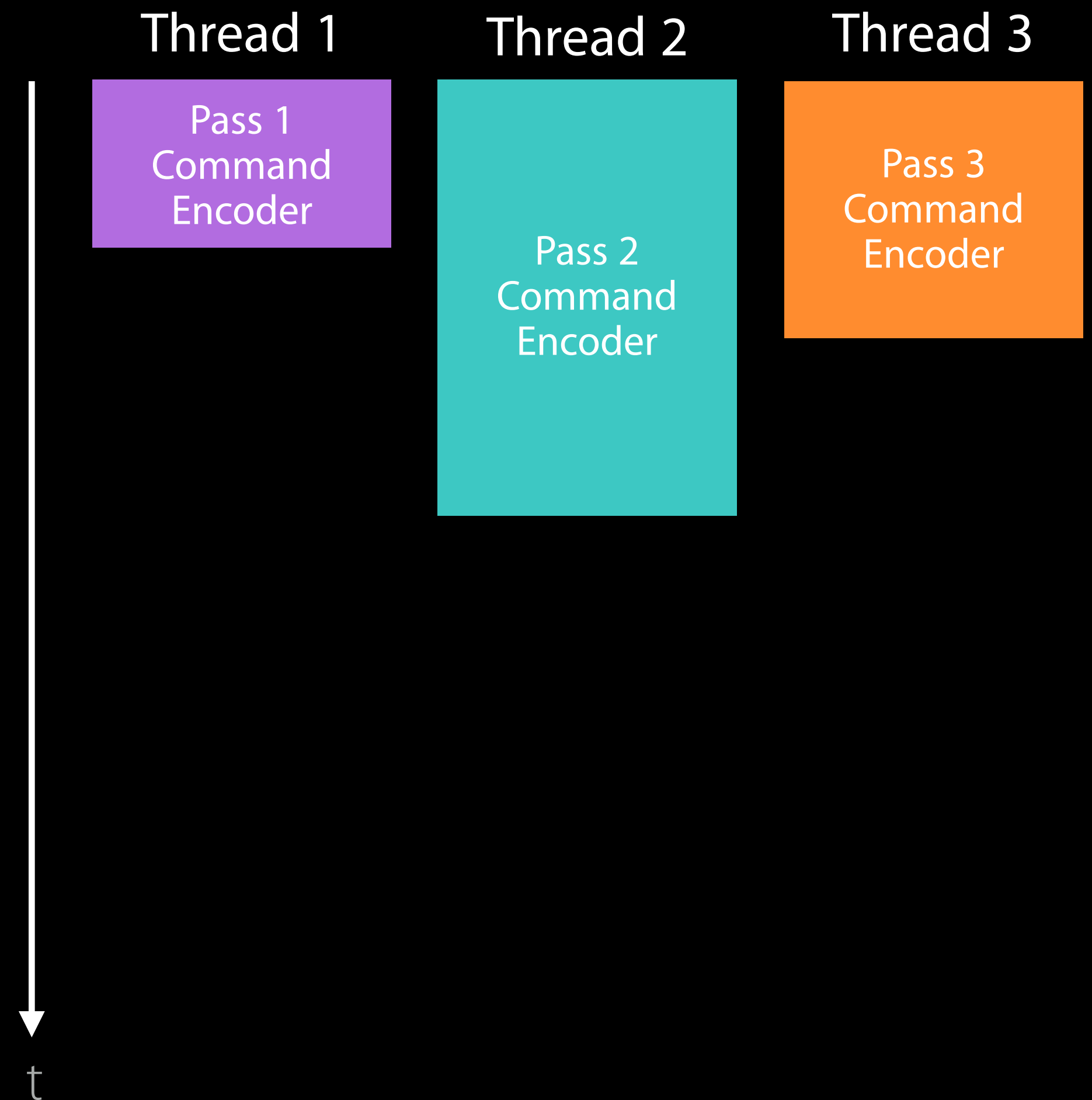
# Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading















# VAIN GLORY

ベイン グローリー











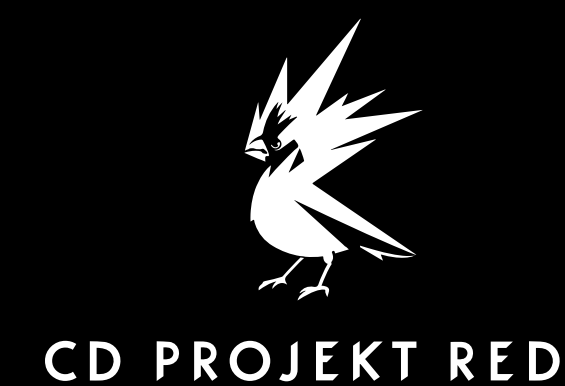
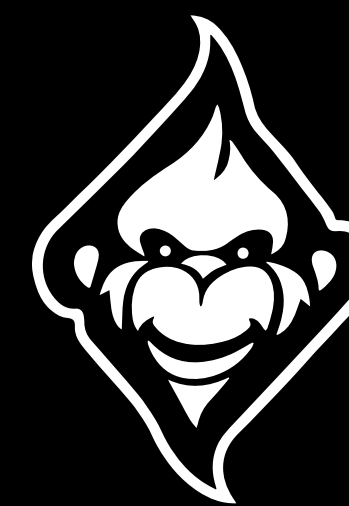
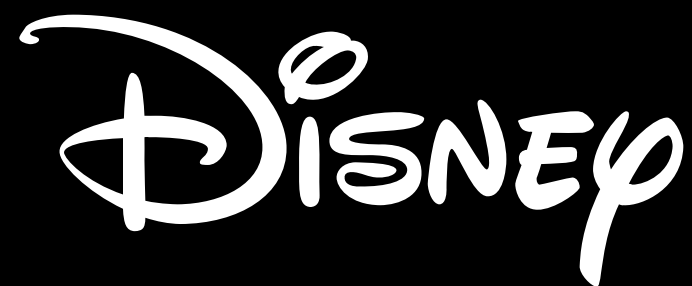
Cancel

Undo

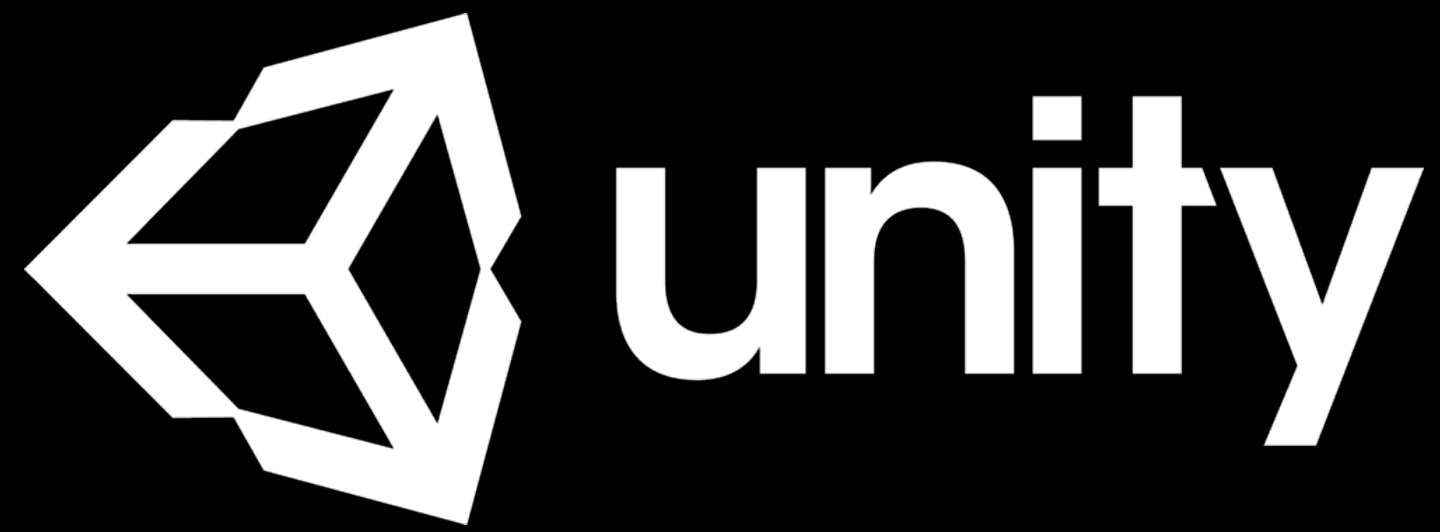
Warp

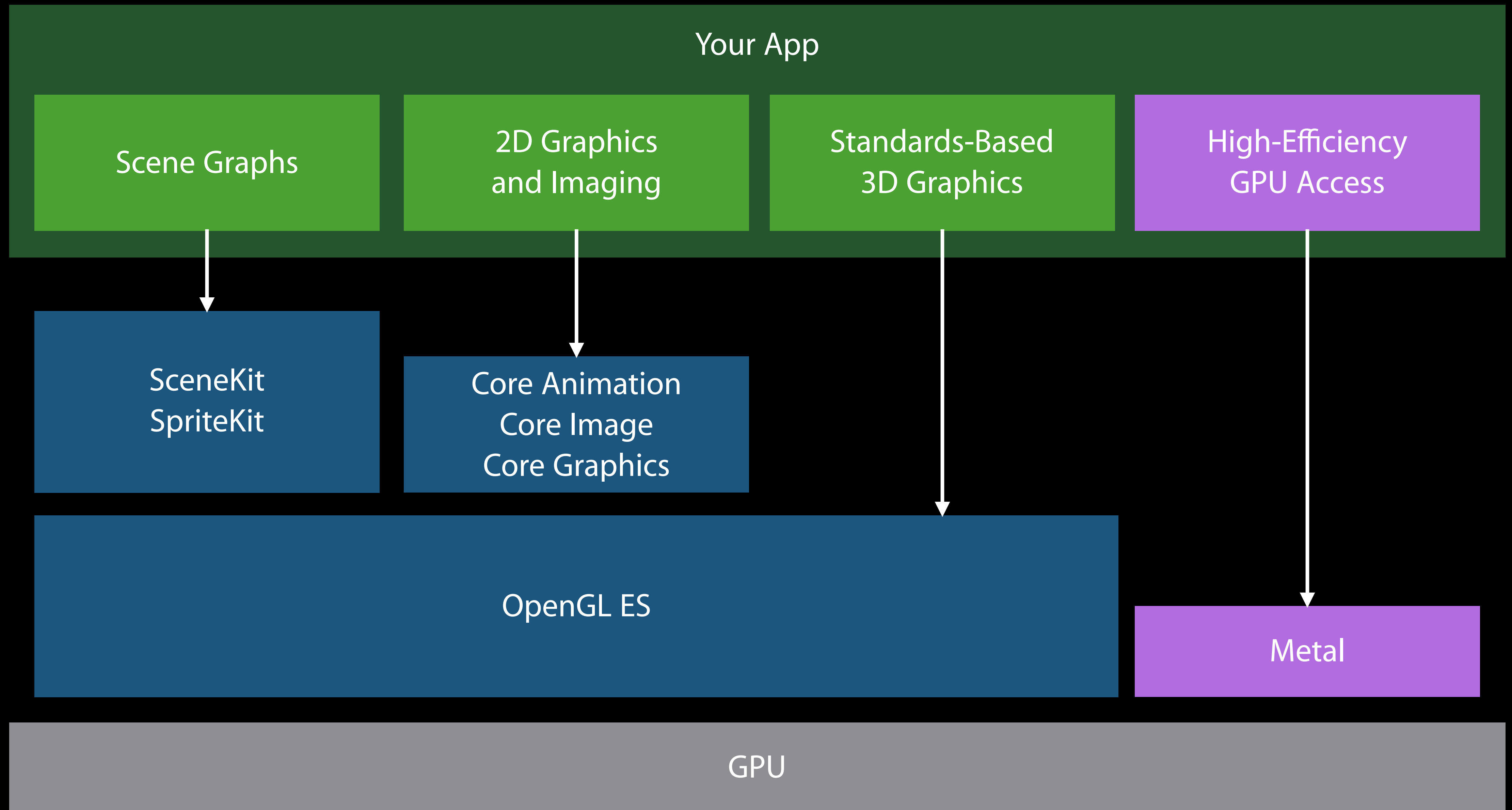
Apply

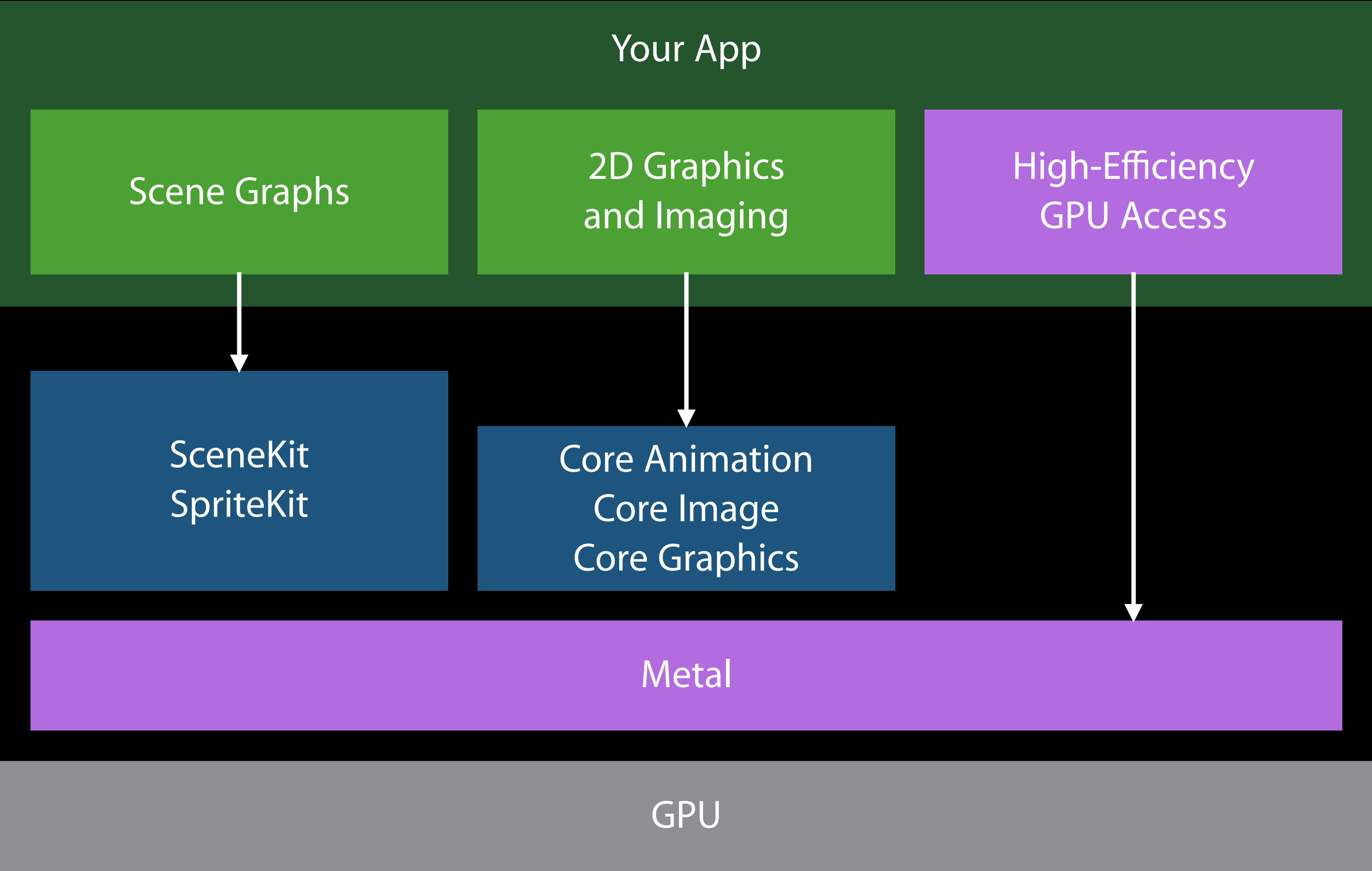


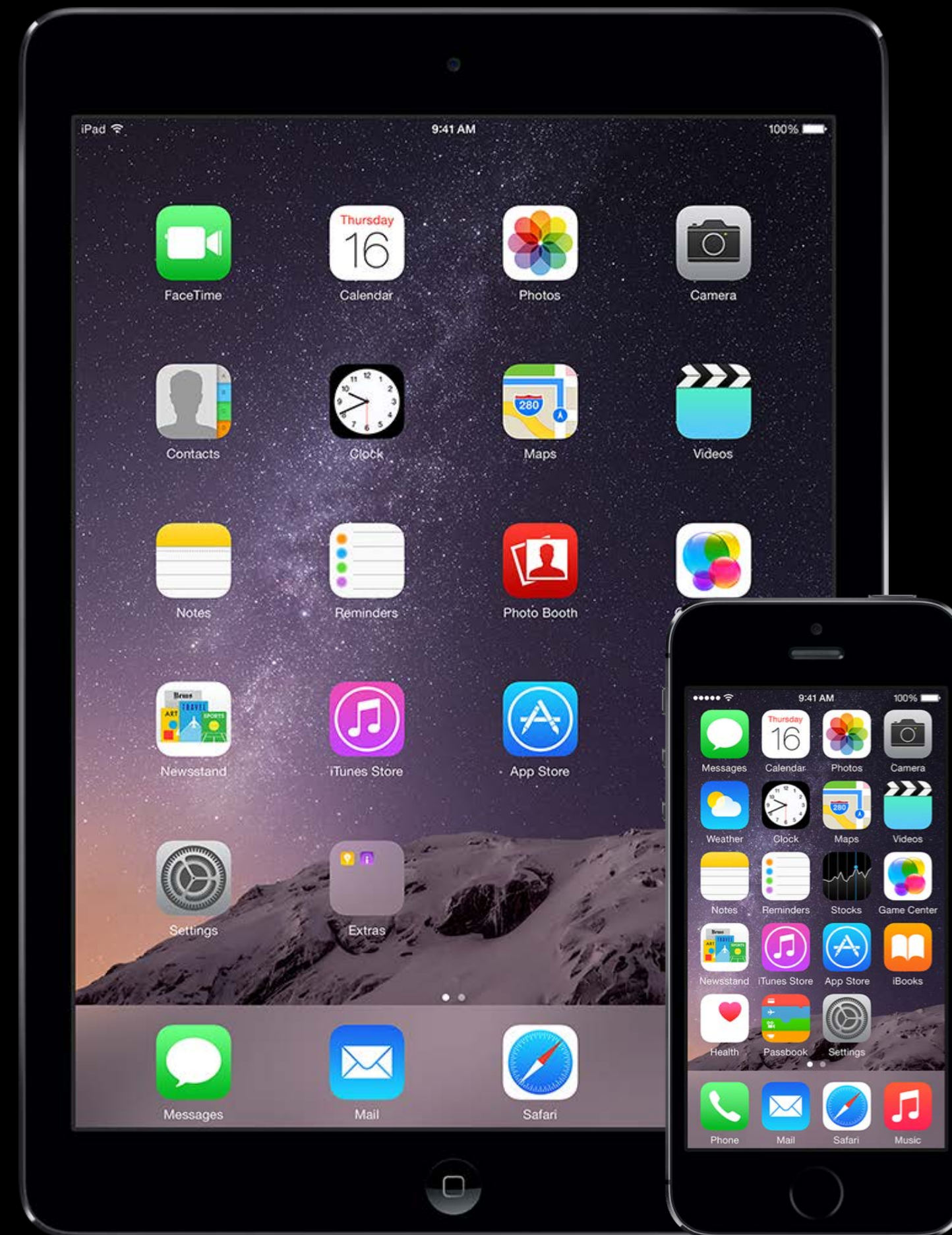
























# Broad Support for Metal





# Tools Support

Frame Debugger

Shader Profiler

Shader Editor

State Inspector

Driver Instruments

API Analysis Tools



Metal OS X

# Metal OS X

Minimal code change required for existing iOS applications

# Metal OS X

Minimal code change required for existing iOS applications

- Device selection



# Metal OS X

Minimal code change required for existing iOS applications

- Device selection
- Support for Discrete Memory

# Metal OS X

Minimal code change required for existing iOS applications

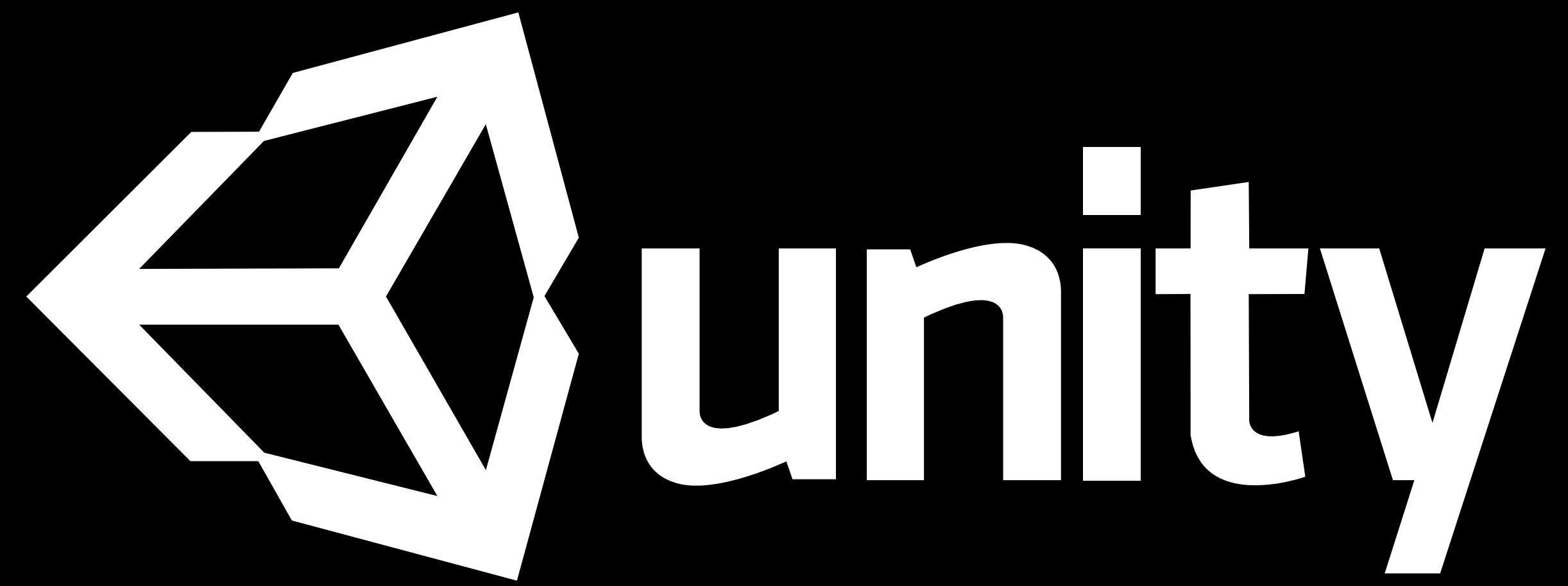
- Device selection
- Support for Discrete Memory
- New texture formats for desktop GPUs













The\_Viking\_Village.unity - trunk - PC, Mac & Linux Standalone

Center Local

Hierarchy

- Content
- Reflection Probes
- WorldBoundaries
- Water
  - Light Probes
  - Occlusion Area
  - Wind Zone
  - Directional light
  - QualityManager
  - Canvas
  - EventSystem
  - WaterReflectionSceneCamera
  - WaterReflection
  - WaterReflection
  - WaterReflection
  - WaterReflection
  - AccessibleVol
  - MouseLock
  - WaterReflection

Scene

Game

Shaded

2D

Gizmos

Inspector

PlayerSettings

Company Name: Unity Technologies

Product Name: Viking Village

Default Icon: None (Texture2D)

Default Cursor: None (Texture2D)

Cursor Hotspot: X 0 Y 0

Settings for PC, Mac & Linux Standalone

Resolution and Presentation

Icon

Splash Image

Other Settings

Rendering

Rendering Path\*: Deferred

Color Space\*: Linear

Automatic Graphics API:  Automatic Graphics API

Automatic Graphics API:  Automatic Graphics API

Reordering the list will switch editor to the first available platform

Graphics APIs for Mac

- Metal
- OpenGL2

Static Batching:

Dynamic Batching:

GPU Skinning\*:

Stereoscopic rendering\*:

Virtual Reality Supported:

Configuration

Scripting Backend: Mono2x

Disable HW Statistics:

Allocated: 1.28 GBOjects: 28088

Viking Village

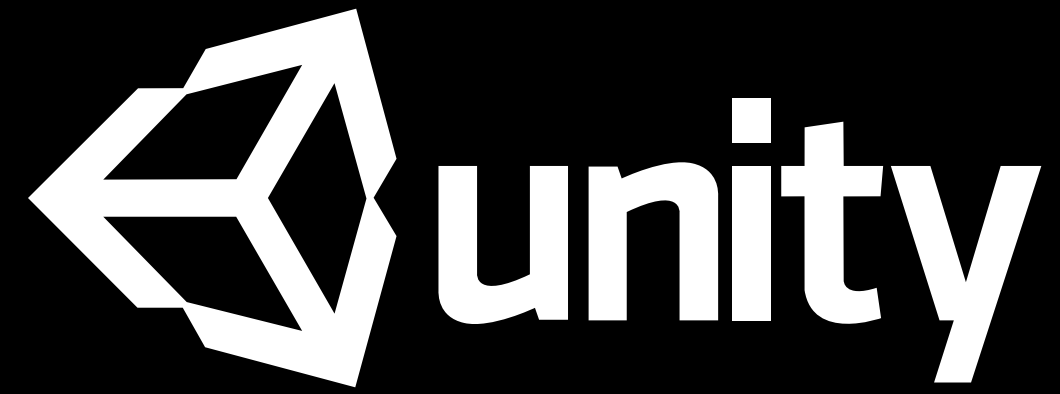
unity

Shaders Standard Ass... Textures

Project

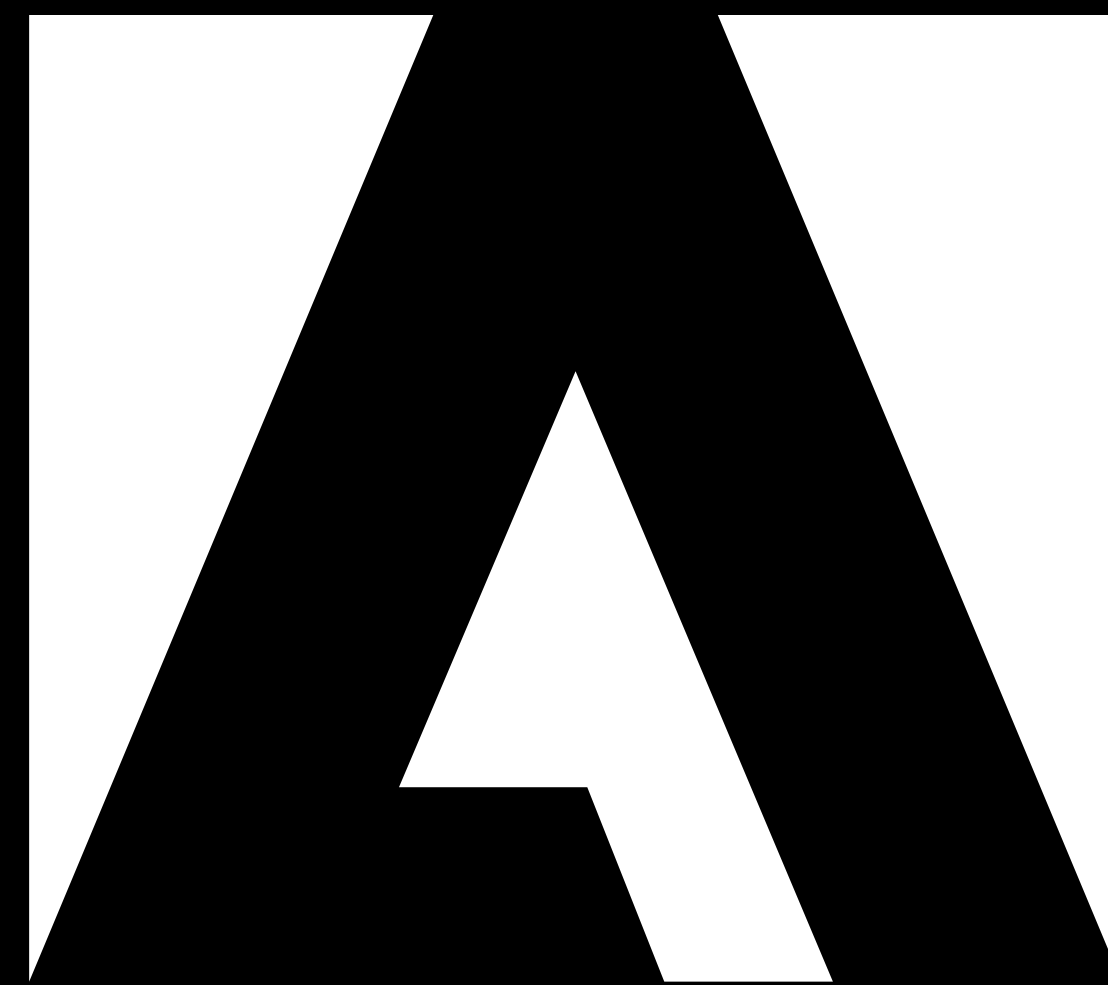
- Assets
  - Animatic
  - Editor
  - Lighting
  - Materials
  - Models
  - Prefabs
  - Scenes
  - Scripts
- ProjectSettings.asset





THE  
FOUNDRY.





**Adobe**

**THE  
FOUNDRY.**



# Adopting Metal on OS X

The Foundry

Jack Greasley





Courtesy of Framestore, Heyday films, and StudioCanal









Courtesy of Adidas









Camera (2)  
Polygons: Face  
Channels: 6  
Deformers: ON  
GL: 1,379,892  
100 mm

Item List

Filter Items Add Item F

- SLS6\_v0011.bxo\*
  - Even Lighting Preset Set
    - Lighting Sphere
    - Reflection Sphere
    - Area Lights
      - Fill
      - Fill (2)
      - Back
      - Key
      - Key (3)
      - Key (2)
    - tx locs
    - Plane (2)
    - SLS
    - Mesh (2)
    - Plane
    - Camera
  - Camera (2)
  - 10X Luminosity (Texture)
  - 007 Red (Texture)
  - 5X Luminosity (Texture)
  - Mesh
  - Texture Group

Shading Channels Info & S... +

View Shader Tree Assign Material F

Filter (none) Add Layer S

Name	Effect
Key Poly (4) (It...	
Back Poly (Item)	(all)
Key Poly (Item)	(all)
Key Poly (2) (It...	
Base Shader	Full Shading
Plane (Item)	(all)
Base Shader ...	Full Shading
Gradient	Group Mask
Scratches (I ...	Driver A
(none) (Image)	Driver A
Material (10)	(all)
Shader (4)	Full Shading
plane (Material)	(all)
Grid	Diffuse Color
Checker	Diffuse Color
Material (25)	(all)
hider (Material)	(all)
Sweep (Material)	(all)
imported shaders	(all)
Group	(all)
Chrome Red ...	(all)
SLS (Item)	(all)

Pass Groups (none) New

Passes (none) New

Auto Add Apply Discard

Properties Groups +

Name Camera (2)

Transform

Position X -2.5131 m  
Y 1.1627 m  
Z 869 mm

Rotation X -8.1452 °  
Y -145.8916 °  
Z 0.0119 °

Order ZXY

Scale X 100.0 %  
Y 100.0 %  
Z 100.0 %

Reset  
Zero  
Add

Set Target

Target Distance 3.0669 m

Sync to View

Projection

Projection Type Perspective

Focal Length 50 mm

Angle of View 39.5978 °

Lens Distortion 0.0

Film Back

Film Back Preset (custom)

Film Width 36 mm  
Height 24 mm  
Film Offset X 0 m  
Y 0 m  
Film Fit Fill

Command

Channels 6

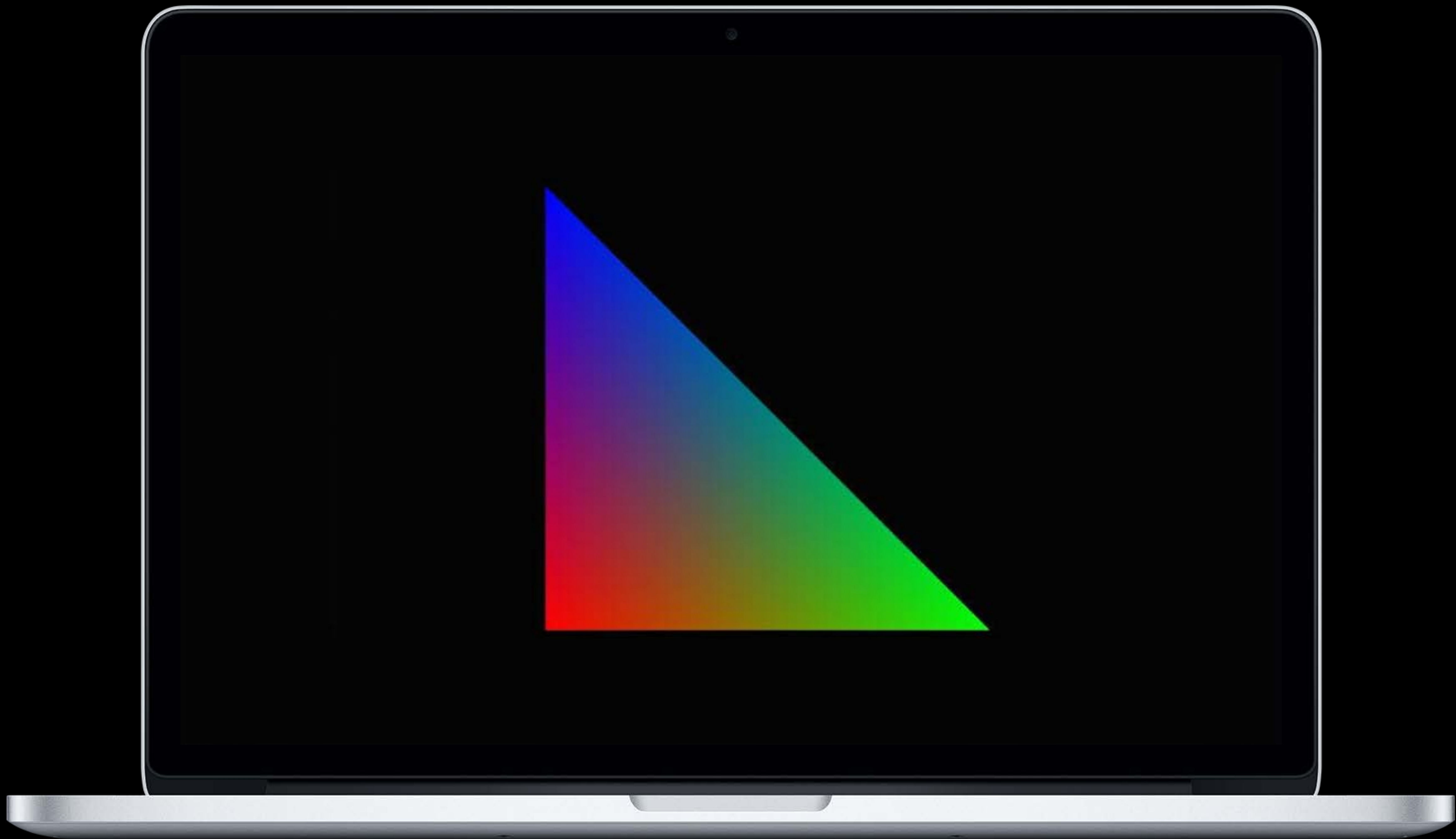
0 12 24 36 48 60 72 84 96 108 120 132 144 156 168 180 192 204 216 228 240 250

Audio Graph Editor Animated 171 Play Play Forwards Settings









# Day One



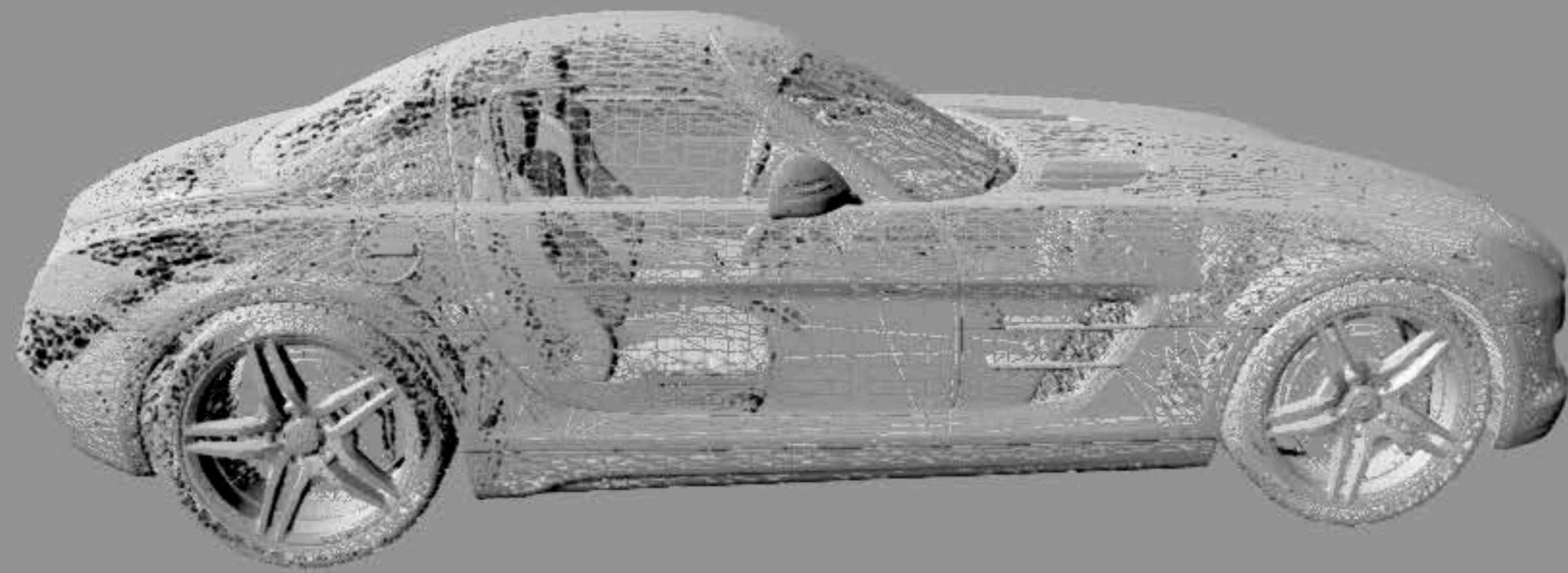


# Day One



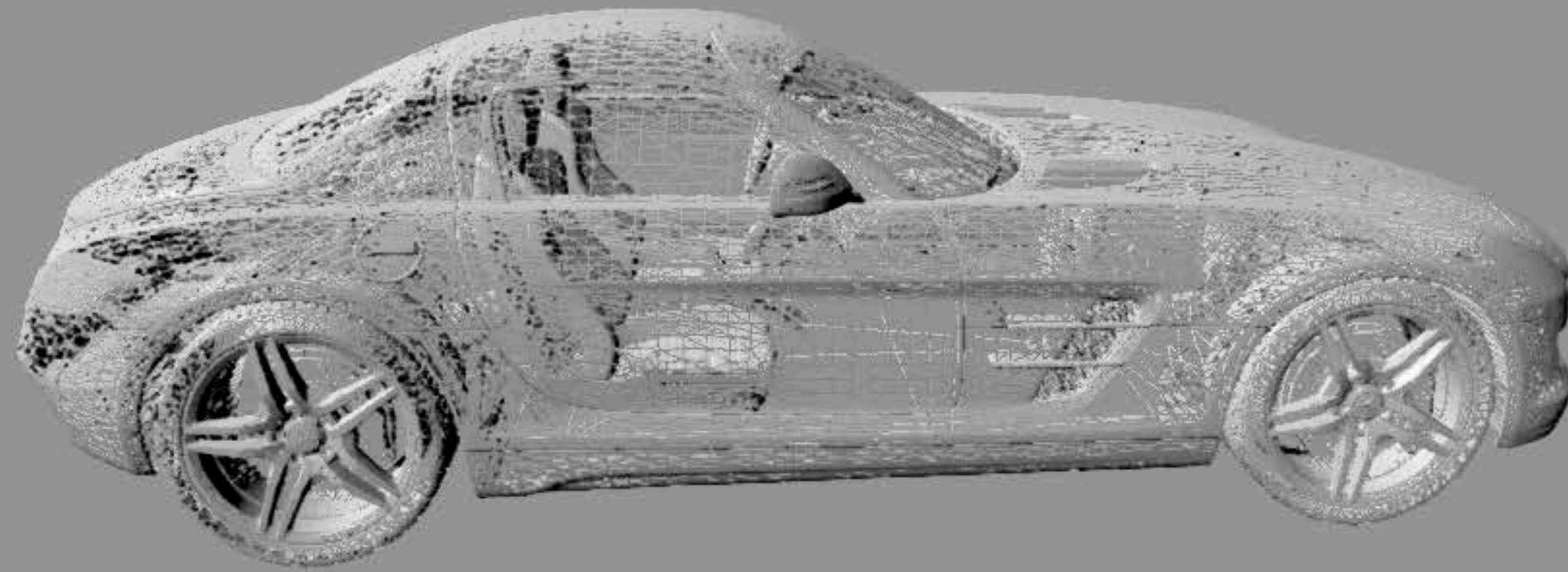


# Day Five



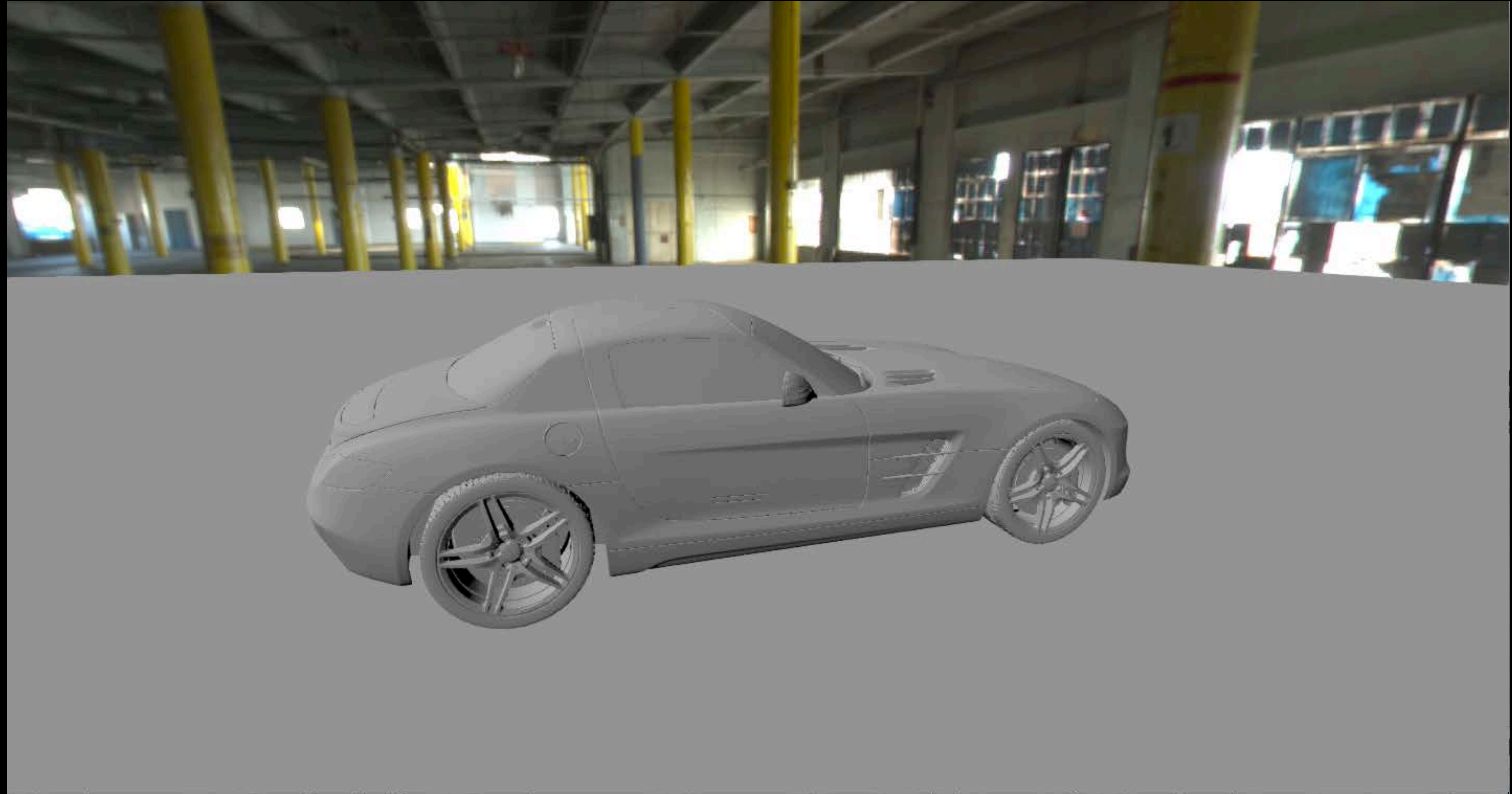


# Day Five



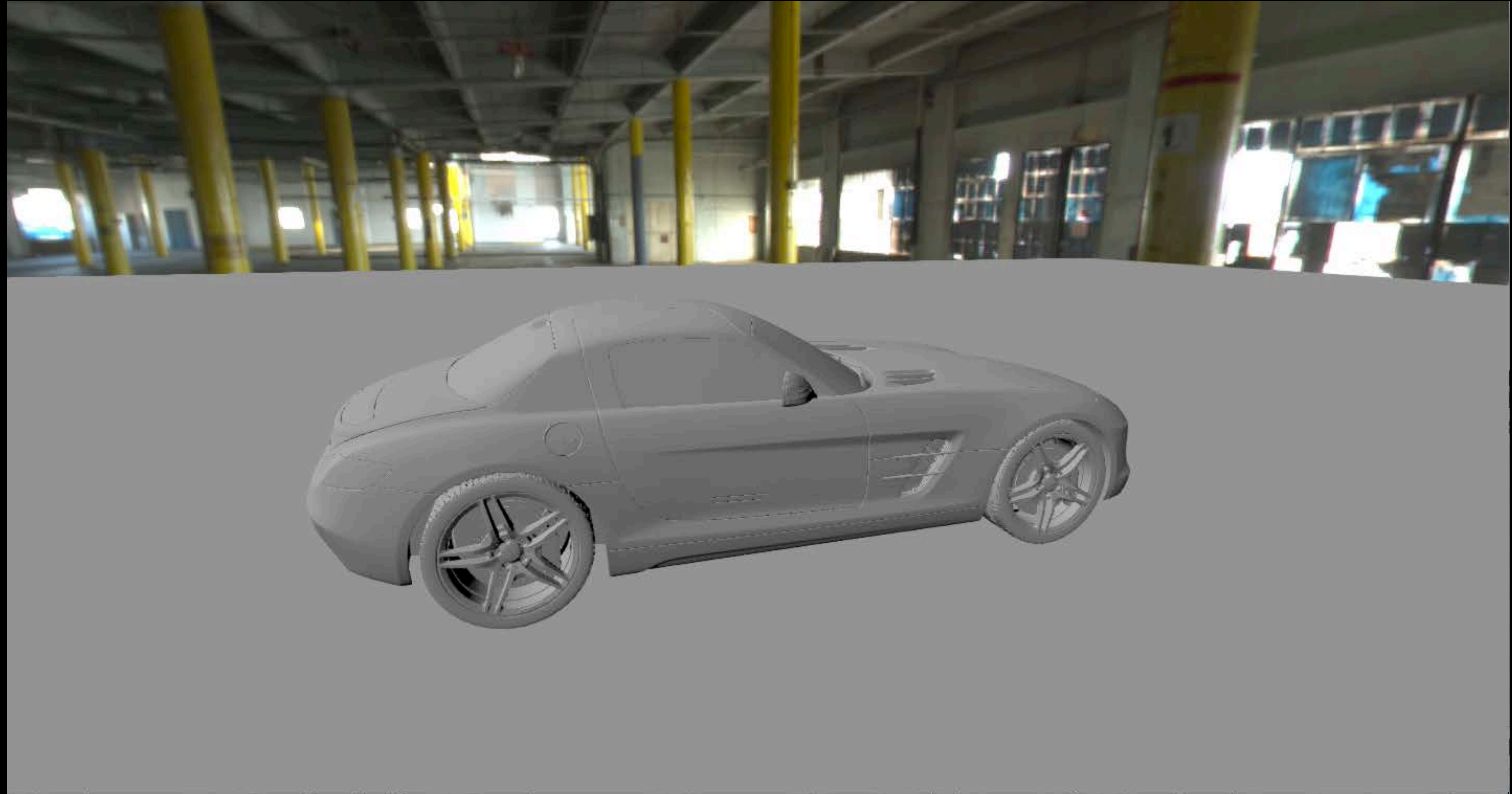


# Day Fifteen

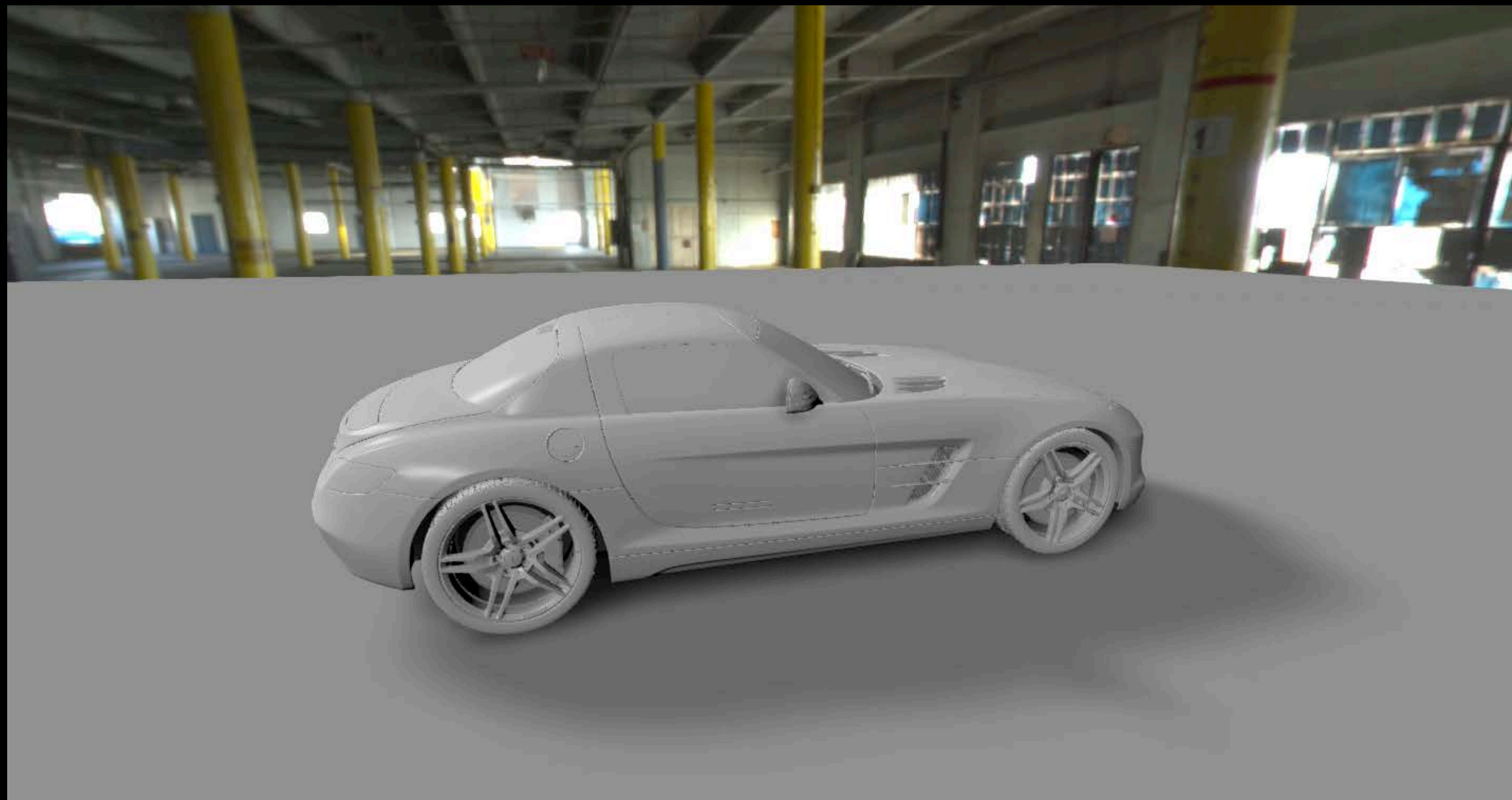




# Day Fifteen

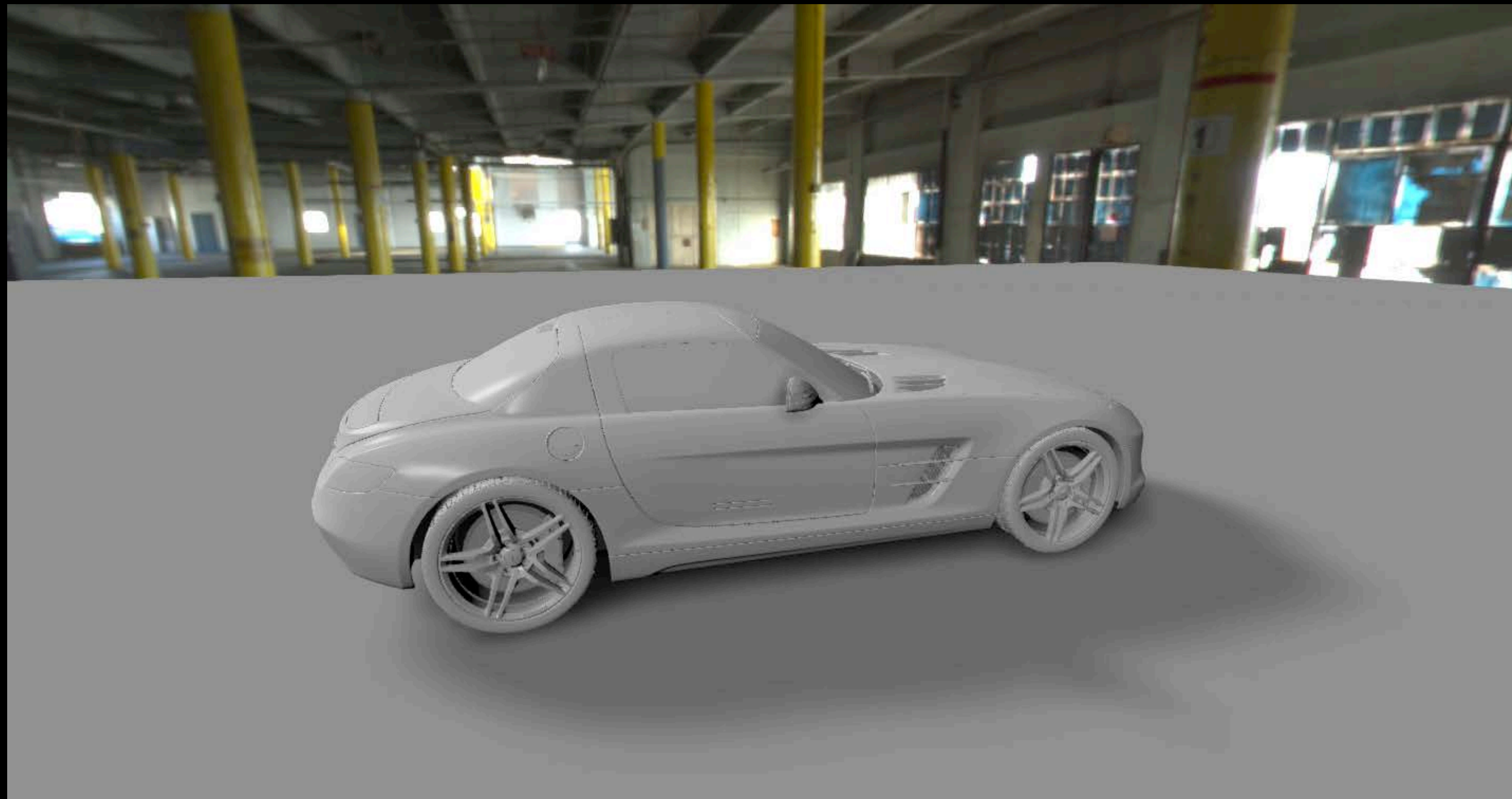


# Day Twenty





# Day Twenty





# Day Twenty-Five

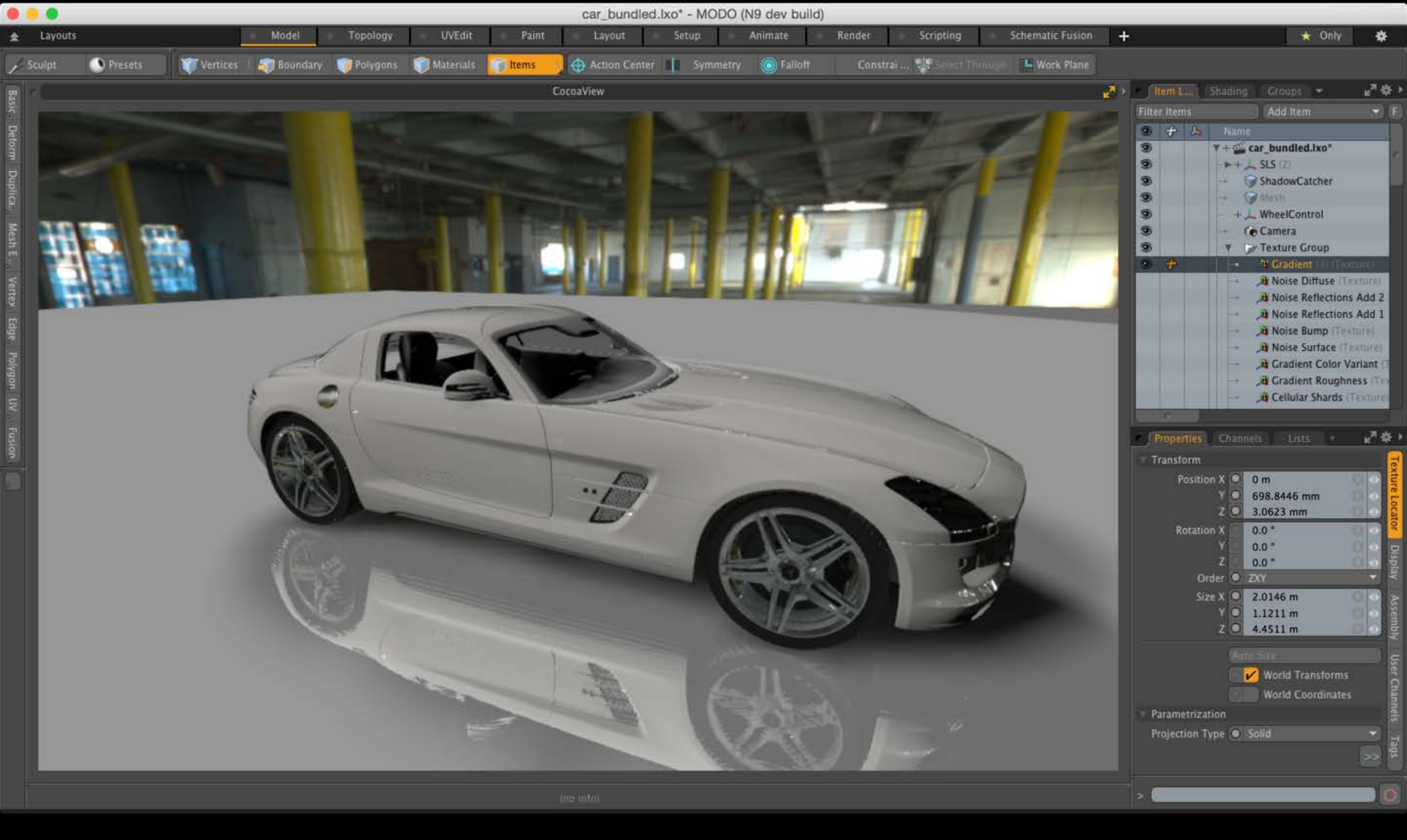




# Day Twenty-Five







CocoaView



Item List | Shading | Groups

Filter Items | Add Item

- car\_bundled.lixo\*
  - SLS (Z)
  - ShadowCatcher
  - Mesh
  - WheelControl
  - Camera
  - Texture Group
    - Gradient (Texture)
      - Noise Diffuse (Texture)
      - Noise Reflections Add 2
      - Noise Reflections Add 1
      - Noise Bump (Texture)
      - Noise Surface (Texture)
      - Gradient Color Variant (Texture)
      - Gradient Roughness (Texture)
      - Cellular Shards (Texture)

Properties | Channels | Lists

Transform

Position X: 0 m  
Y: 698.8446 mm  
Z: 3.0623 mm

Rotation X: 0.0 °  
Y: 0.0 °  
Z: 0.0 °

Order: ZXY

Size X: 2.0146 m  
Y: 1.1211 m  
Z: 4.4511 m

Auto Size

World Transforms  
 World Coordinates

Parametrization

Projection Type: Solid

Texture Locator | Display | Assembly | User Channels | Tags



What did we learn?

New Features





Device Selection

New Compressed Texture Formats

Texture Barriers

New Texture Features

MetalKit

Layer Select

Metal Performance Shaders

Counting Occlusion Queries



New Memory Model

Multi-sample Depth resolves

Depth Clipping Support

GPU Family Sets

Draw and Compute Indirect

Metal System Trace Tool

Separate front/back stencil reference values

New Shader Constant Update APIs



Device Selection

New Compressed Texture Formats

Texture Barriers

New Texture Features

MetalKit

Layer Select

Metal Performance Shaders

Counting Occlusion Queries



New Memory Model

Multi-sample Depth resolves

Depth Clipping Support

GPU Family Sets

Draw and Compute Indirect

Metal System Trace

Separate front/back stencil reference values

New Shader Constant Update APIs

Device Selection

New Compressed Texture Formats

Texture Barriers

New Texture Features

MetalKit

Layer Select

Metal Performance Shaders

Counting Occlusion Queries

New Memory Model

Multi-sample Depth resolves



Depth Clipping Support

GPU Family Sets

Draw and Compute Indirect

Metal System Trace Tool

Separate front/back stencil reference values

New Shader Constant Update APIs



# Metal Feature Set Definitions

# Metal Feature Set Definitions

Feature sets represent a collection of capabilities by GPU generation

`iOS_GPUFamily2_v1`



# Metal Feature Set Definitions

Feature sets represent a collection of capabilities by GPU generation

- Prefix defines the platform

**iOS**\_GPUFamily2\_v1

# Metal Feature Set Definitions

Feature sets represent a collection of capabilities by GPU generation

- Prefix defines the platform
- Family Name is unique to a hardware generation

iOS\_GPUFamily2\_v1



# Metal Feature Set Definitions

Feature sets represent a collection of capabilities by GPU generation

- Prefix defines the platform
- Family Name is unique to a hardware generation
- Revision number can change as features are added over time

iOS\_GPUFamily2\_v1

# Metal Feature Set Definitions

Simple query to see if device supports a given feature set



# Metal Feature Set Definitions

Simple query to see if device supports a given feature set

```
[myMetalDevice supportsFeatureSet:iOS_GPUFamily2_v1]
```

# iOS Metal Feature Sets

Name	Introduced	Feature Additions	Supported Devices
iOS_GPUFamily1_v1	iOS 8	Core Metal features for A7 devices	iPhone 5s
iOS_GPUFamily1_v2	iOS 9	New texture features Multi-sample depth resolves Depth clipping support Separate stencil reference values	iPad Air iPhone 6 and 6 Plus iPad Air 2
iOS_GPUFamily2_v1	iOS 8	ASTC texture support	
iOS_GPUFamily2_v2	iOS 9	New texture features Multi-sample depth resolves Depth clipping support Separate stencil reference values	iPhone 6 and 6 Plus iPad Air 2

# OS X Metal Feature Sets

Name	Introduced	Feature Additions	Supported Devices
OSX_GPUFamily1_v1	OS X 10.11	Same core feature as iOS, plus BCn texture compression formats Combined depth stencil formats Managed resource model Multi-GPU device selection Draw and compute indirect counting occlusion queries Layer select Texture barriers New texture features Multi-sample depth resolves Depth clipping support Separate stencil reference values	All Macs since 2012



# Shader Constant Updates

# Shader Constant Updates

Command Buffer

Draw 1

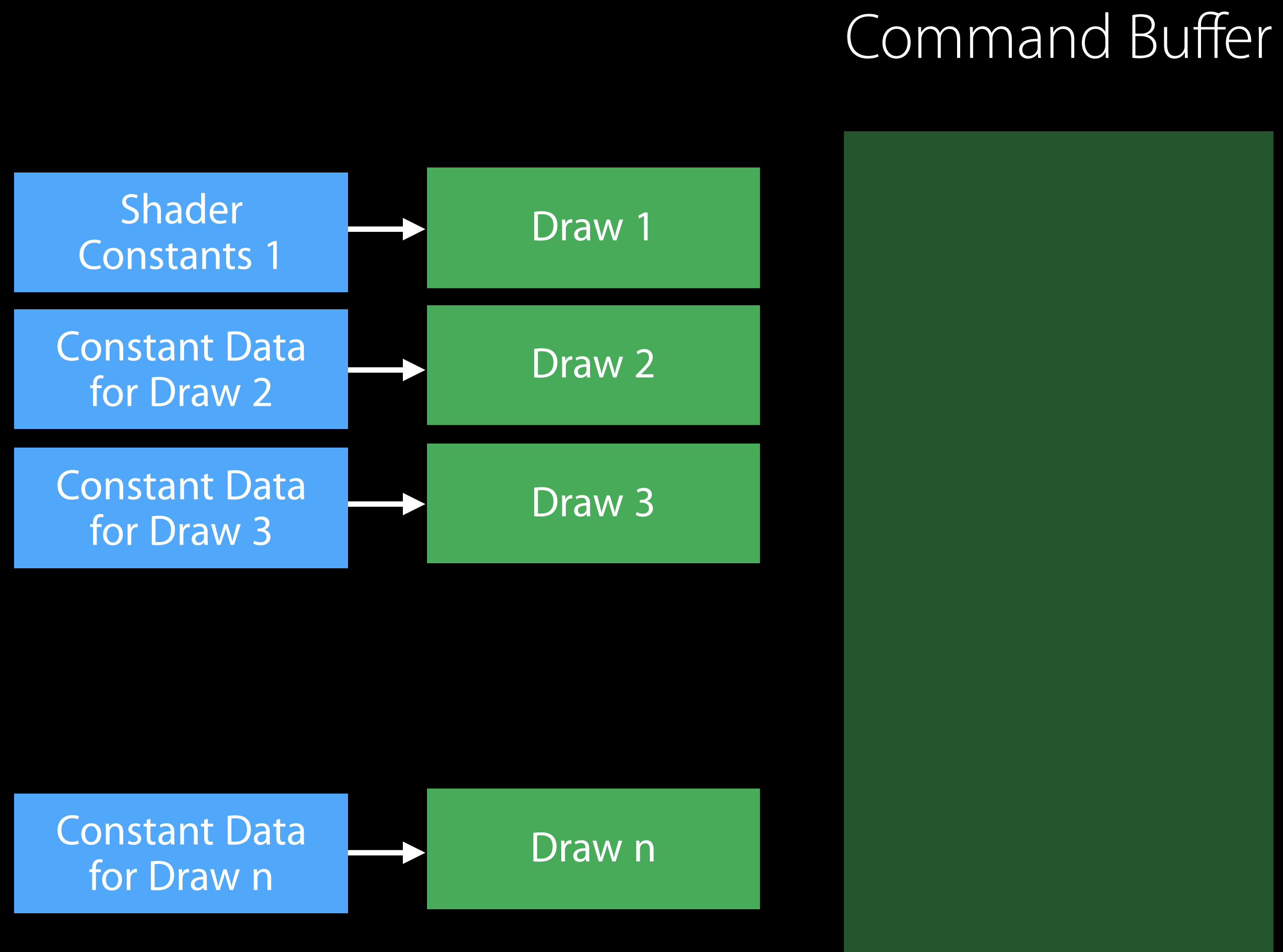
Draw 2

Draw 3

Draw n



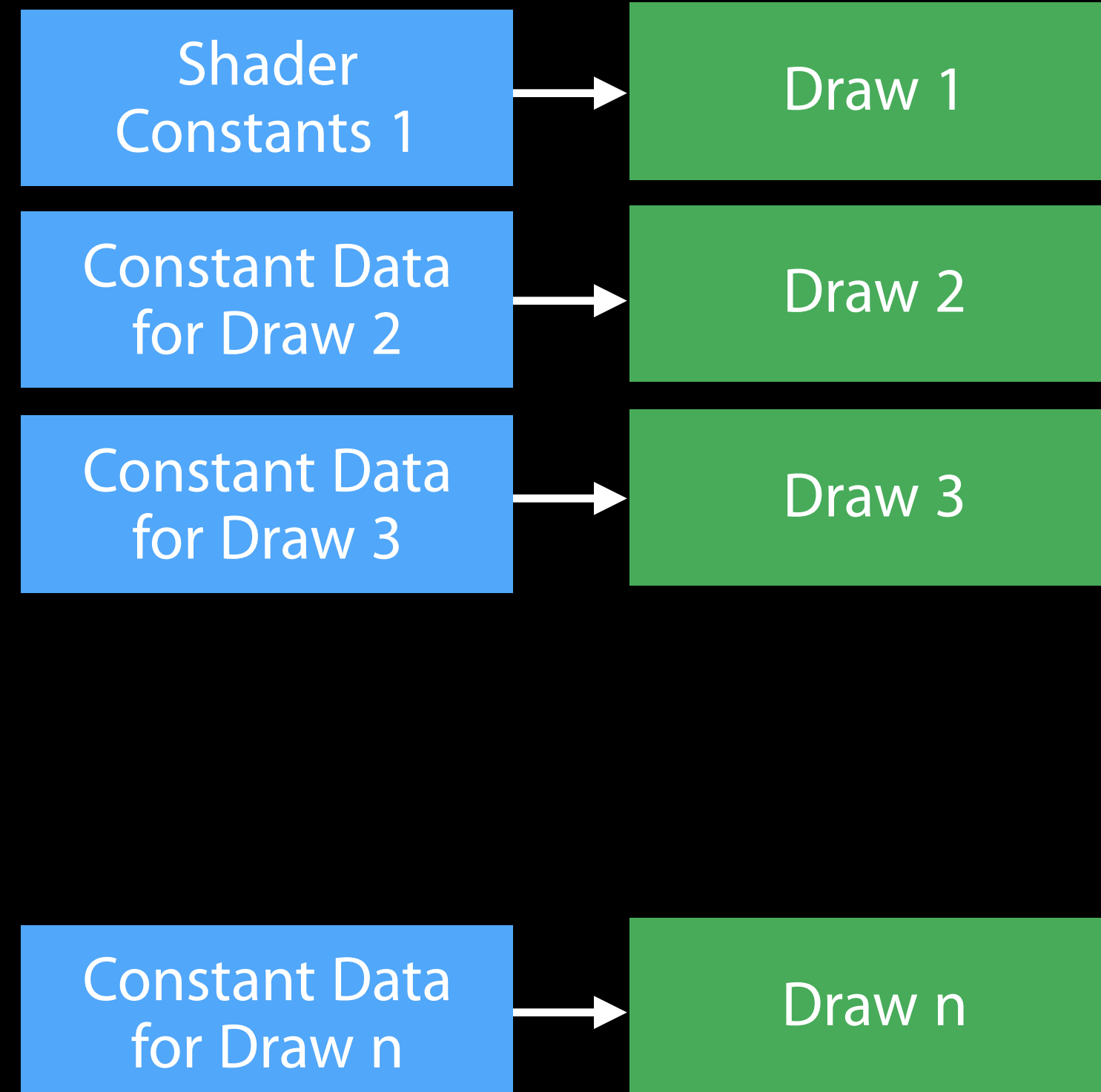
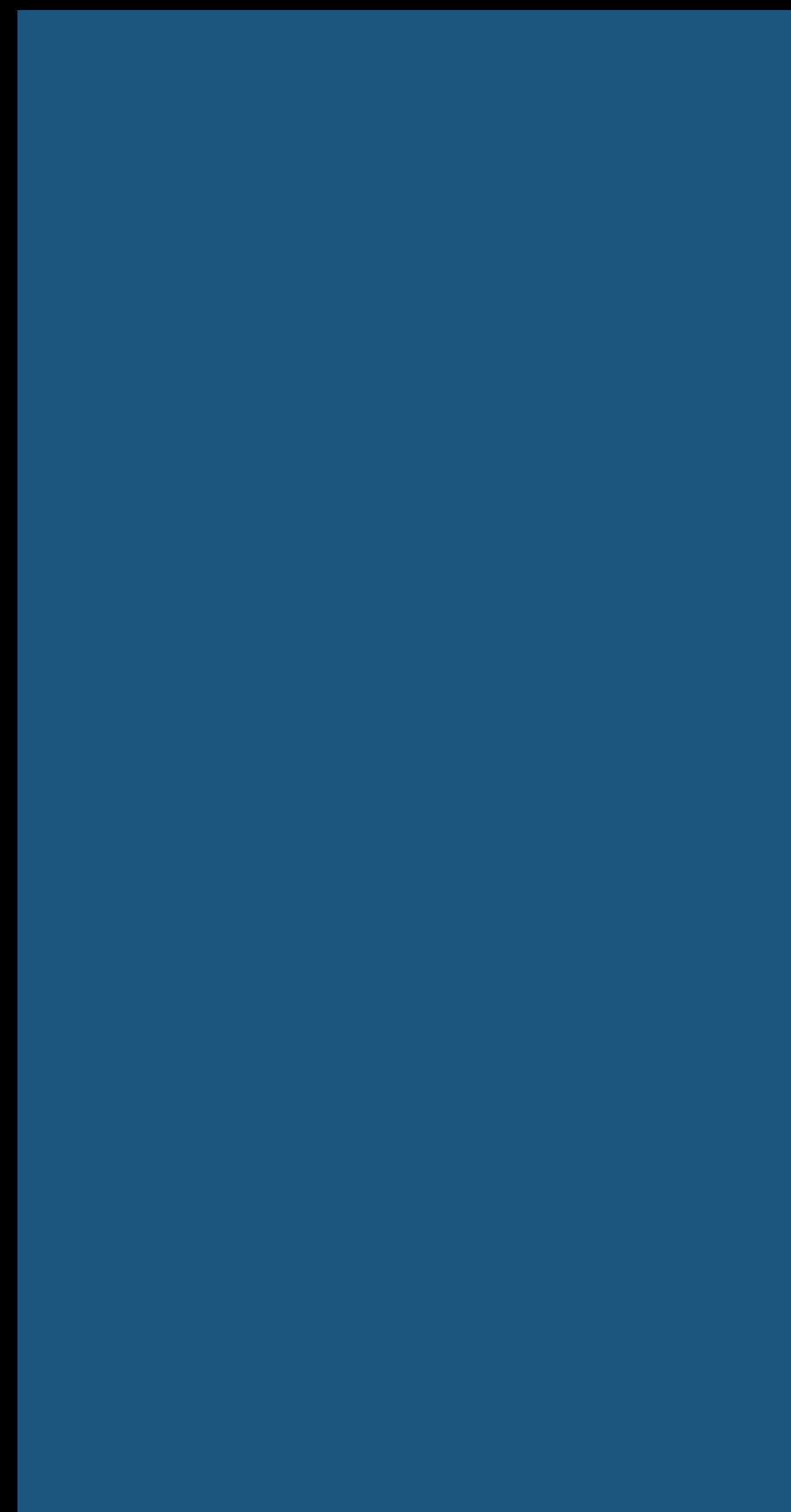
# Shader Constant Updates





# Shader Constant Updates

Constant Buffer

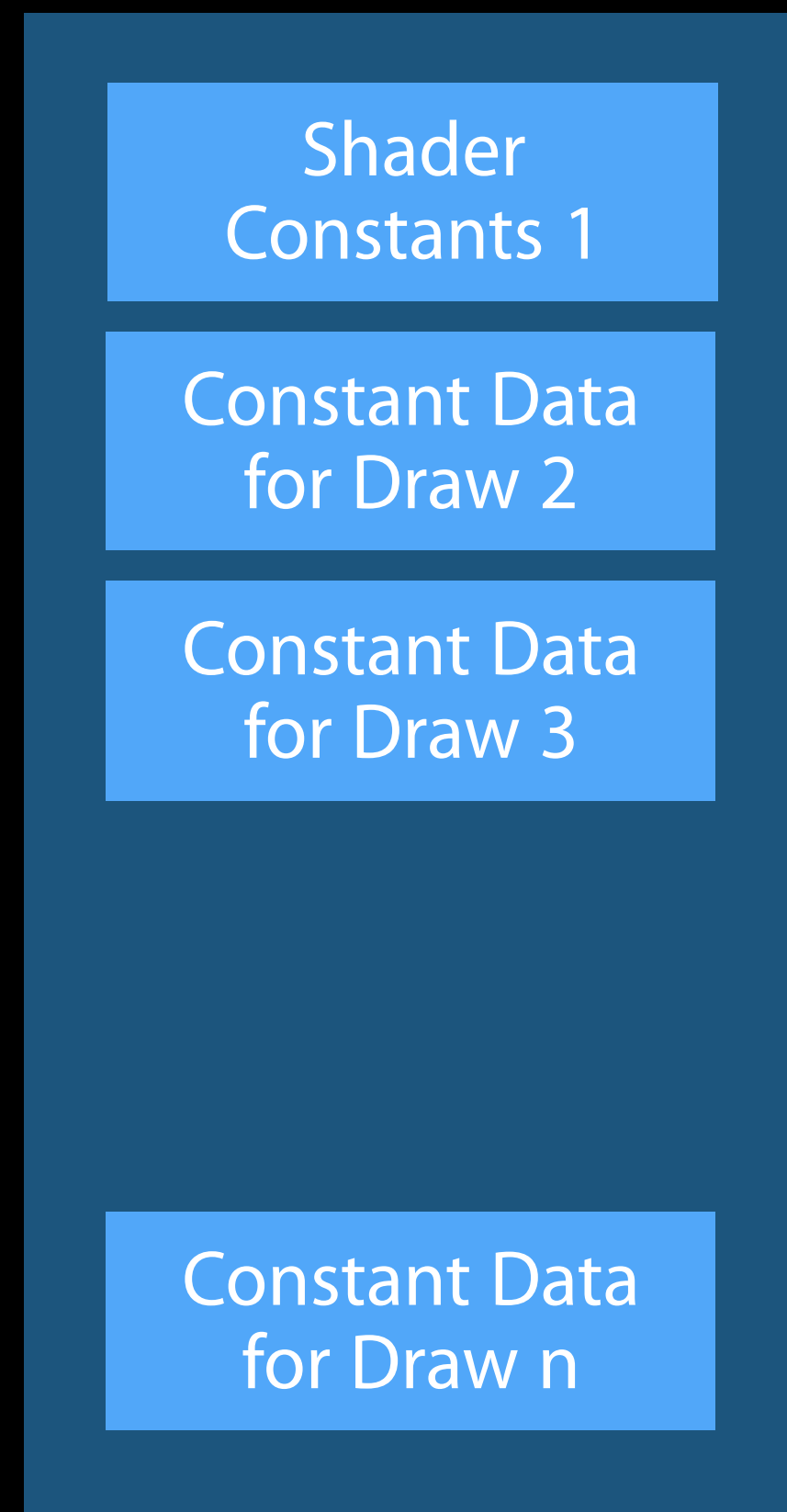


Command Buffer

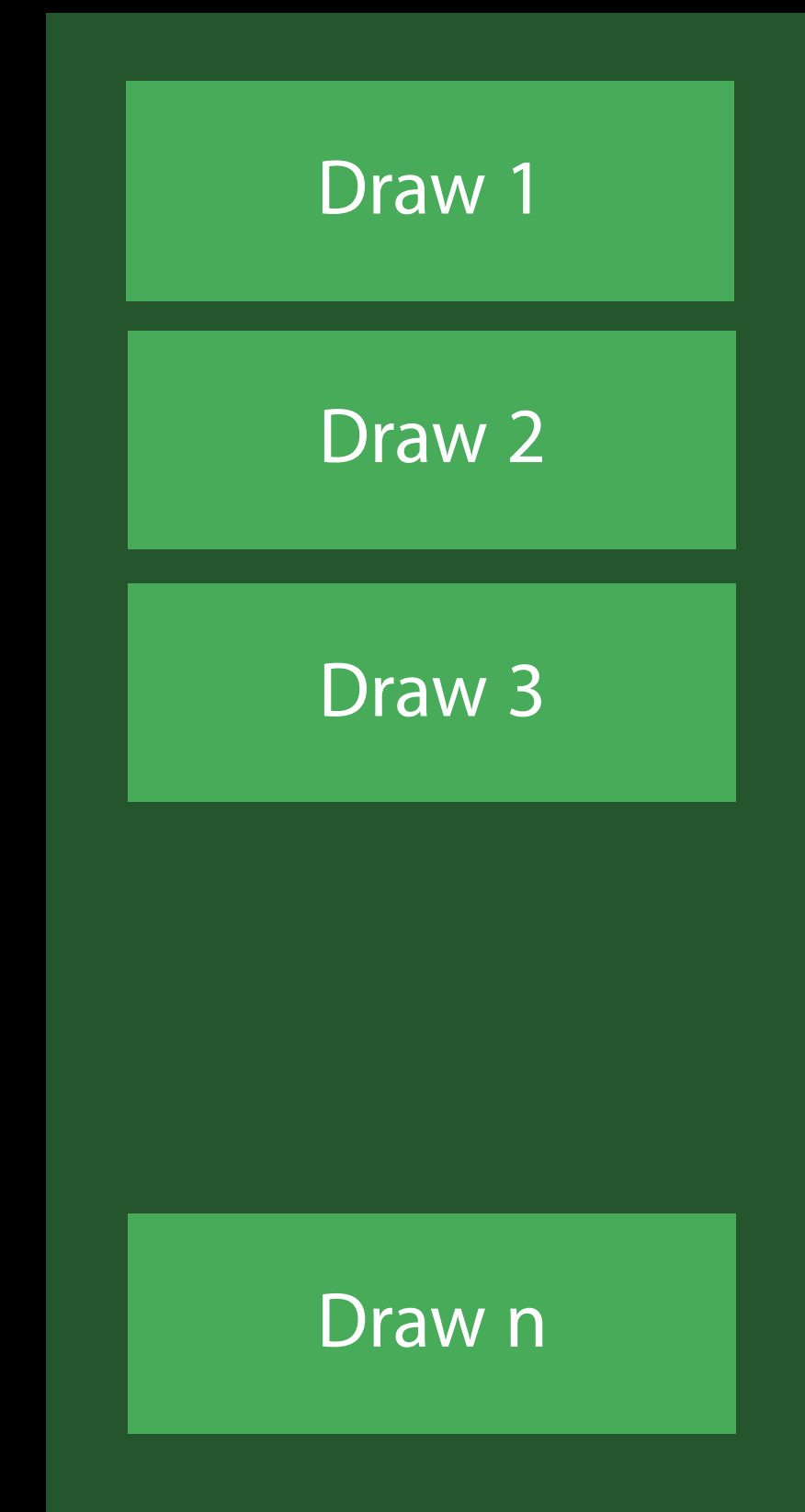


# Shader Constant Updates

## Constant Buffer



## Command Buffer



# Shader Constant Updates

Bind per draw

```
id <MTLBuffer> constant_buffer = ...;
MyConstants* constant_ptr = constant_buffer.contents;

for (i=0; i<draw_count; i++)
{
    constant_ptr[i] = // write constants directly into the buffer

    [render_pass setVertexBuffer:constant_buffer offset:i*sizeof(MyConstants) atIndex:0];

    // draw
}
```



# Shader Constant Updates

Bind per draw

```
id <MTLBuffer> constant_buffer = ...;  
MyConstants* constant_ptr = constant_buffer.contents;
```

```
for (i=0; i<draw_count; i++)  
{  
    constant_ptr[i] = // write constants directly into the buffer  
  
    [render_pass setVertexBuffer:constant_buffer offset:i*sizeof(MyConstants) atIndex:0];  
  
    // draw  
}
```

# Shader Constant Updates

Bind per draw

```
id <MTLBuffer> constant_buffer = ...;
MyConstants* constant_ptr = constant_buffer.contents;

for (i=0; i<draw_count; i++)
{
    constant_ptr[i] = // write constants directly into the buffer

    [render_pass setVertexBuffer:constant_buffer offset:i*sizeof(MyConstants) atIndex:0];

    // draw
}
```

# Shader Constant Updates

Bind per draw

```
id <MTLBuffer> constant_buffer = ...;
MyConstants* constant_ptr = constant_buffer.contents;

for (i=0; i<draw_count; i++)
{
    constant_ptr[i] = // write constants directly into the buffer

    [render_pass setVertexBuffer:constant_buffer offset:i*sizeof(MyConstants) atIndex:0];

    // draw
}
```



# Shader Constant Updates

Bind per draw

```
id <MTLBuffer> constant_buffer = ...;
MyConstants* constant_ptr = constant_buffer.contents;

for (i=0; i<draw_count; i++)
{
    constant_ptr[i] = // write constants directly into the buffer

    [render_pass setVertexBuffer:constant_buffer offset:i*sizeof(MyConstants) atIndex:0];

    // draw
}
```

# Shader Constant Updates

Bind per draw

```
id <MTLBuffer> constant_buffer = ...;
MyConstants* constant_ptr = constant_buffer.contents;

for (i=0; i<draw_count; i++)
{
    constant_ptr[i] = // write constants directly into the buffer

    [render_pass setVertexBuffer:constant_buffer offset:i*sizeof(MyConstants) atIndex:0];

    // draw
}
```

# Shader Constant Updates

Bind per draw

```
id <MTLBuffer> constant_buffer = ...;
MyConstants* constant_ptr = constant_buffer.contents;

for (i=0; i<draw_count; i++)
{
    constant_ptr[i] = // write constants directly into the buffer
    [renderpass setVertexBuffer: constant_buffer offset: 0 atIndex: 0];
    [renderpass setVertexBufferOffset: i*sizeof(MyConstants) atIndex: 0];

    // draw
}
```



# Shader Constant Updates

Bind per draw

```
id <MTLBuffer> constant_buffer = ...;
MyConstants* constant_ptr = constant_buffer.contents;

for (i=0; i<draw_count; i++)
{
    constant_ptr[i] = // write constants directly into the buffer
    [renderpass setVertexBuffer: constant_buffer offset: 0 atIndex: 0];
    [renderpass setVertexBufferOffset: i*sizeof(MyConstants) atIndex: 0];

    // draw
}
```

# Shader Constant Updates

Bind per draw

```
id <MTLBuffer> constant_buffer = ...;
MyConstants* constant_ptr = constant_buffer.contents;

for (i=0; i<draw_count; i++)
{
    constant_ptr[i] = // write constants directly into the buffer
    [renderpass setVertexBuffer: constant_buffer offset: 0 atIndex: 0];
    [renderpass setVertexBufferOffset:i*sizeof(MyConstants) atIndex:0];

    // draw
}
```

# Shader Constant Updates

Bind per draw

```
id <MTLBuffer> constant_buffer = ...;
MyConstants* constant_ptr = constant_buffer.contents;

for (i=0; i<draw_count; i++)
{
    constant_ptr[i] = // write constants directly into the buffer
    [renderpass setVertexBuffer: constant_buffer offset: 0 atIndex: 0];
    [renderpass setVertexBufferOffset:i*sizeof(MyConstants) atIndex:0];

    // draw
}
```



# Shader Constant Updates

Bind per draw

```
id <MTLBuffer> constant_buffer = ...;
MyConstants* constant_ptr = constant_buffer.contents;

[renderpass setVertexBuffer: constant_buffer offset: 0 atIndex: 0];

for (i=0; i<draw_count; i++)
{
    constant_ptr[i] = // write constants directly into the buffer

    [renderpass setVertexBufferOffset:i*sizeof(MyConstants) atIndex:0];

    // draw
}
```

# Shader Constant Updates

Internal constant buffer managed by Metal

```
for (i=0; i<draw_count; i++)
{
    MyConstants constants = // generate constants onto the stack
    [renderpass setVertexBytes:&constants length:sizeof(MyConstants) atIndex:0];

    // draw
}
```

# Shader Constant Updates

Internal constant buffer managed by Metal

```
for (i=0; i<draw_count; i++)  
{  
    MyConstants constants = // generate constants onto the stack  
    [renderpass setVertexBytes:&constants length:sizeof(MyConstants) atIndex:0];  
  
    // draw  
}
```



# New Memory Model

# New Memory Model

Support both unified and Discrete Memory model with minimal code change

# New Memory Model

Support both unified and Discrete Memory model with minimal code change

New storage modes to specify where the resource should reside

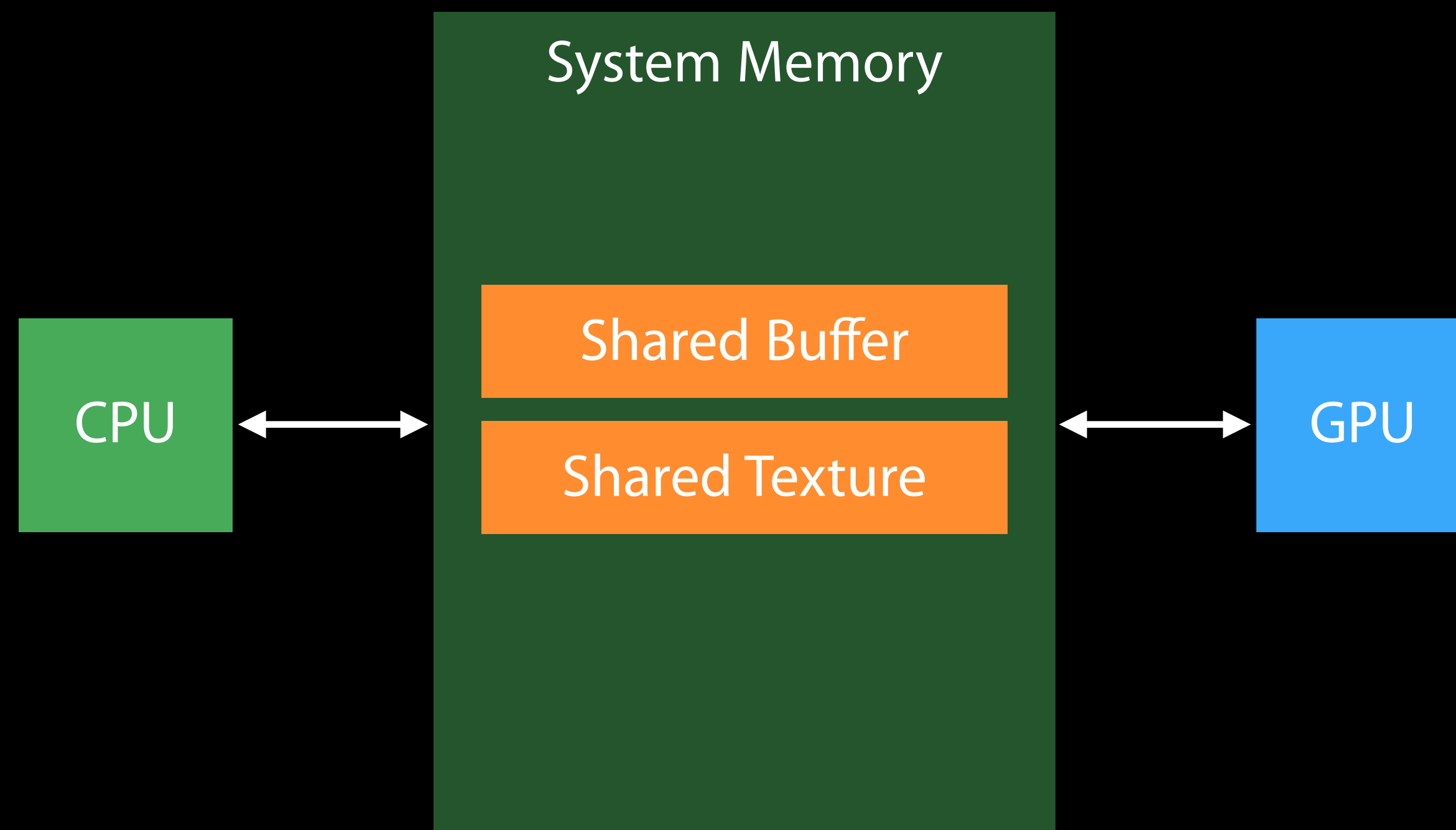
- Shared storage mode
- Private storage mode
- Managed storage mode



# Shared Storage Mode

Introduced with iOS 8

Full coherency between CPU and GPU at command buffer boundaries

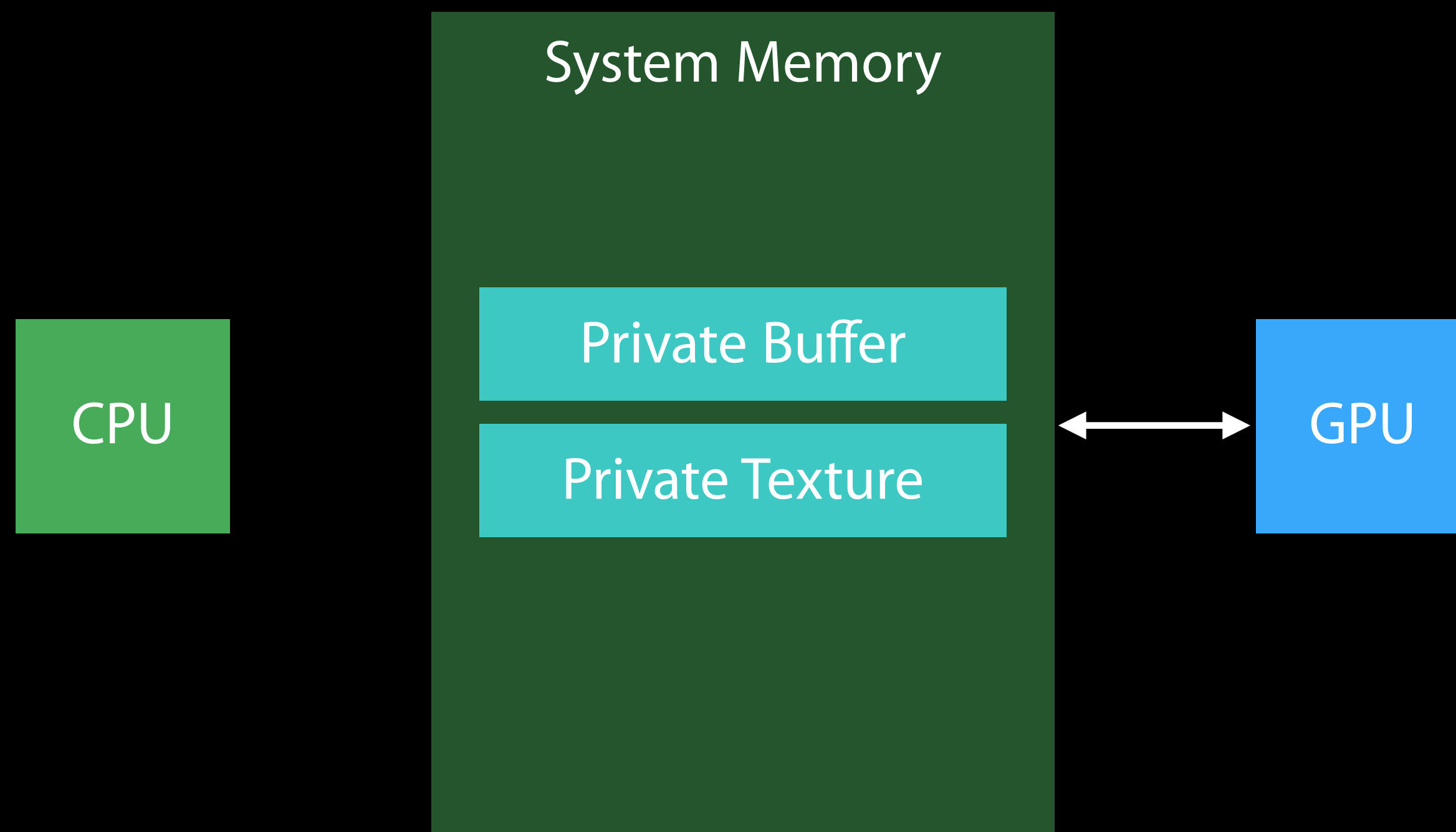


# Private Storage Mode

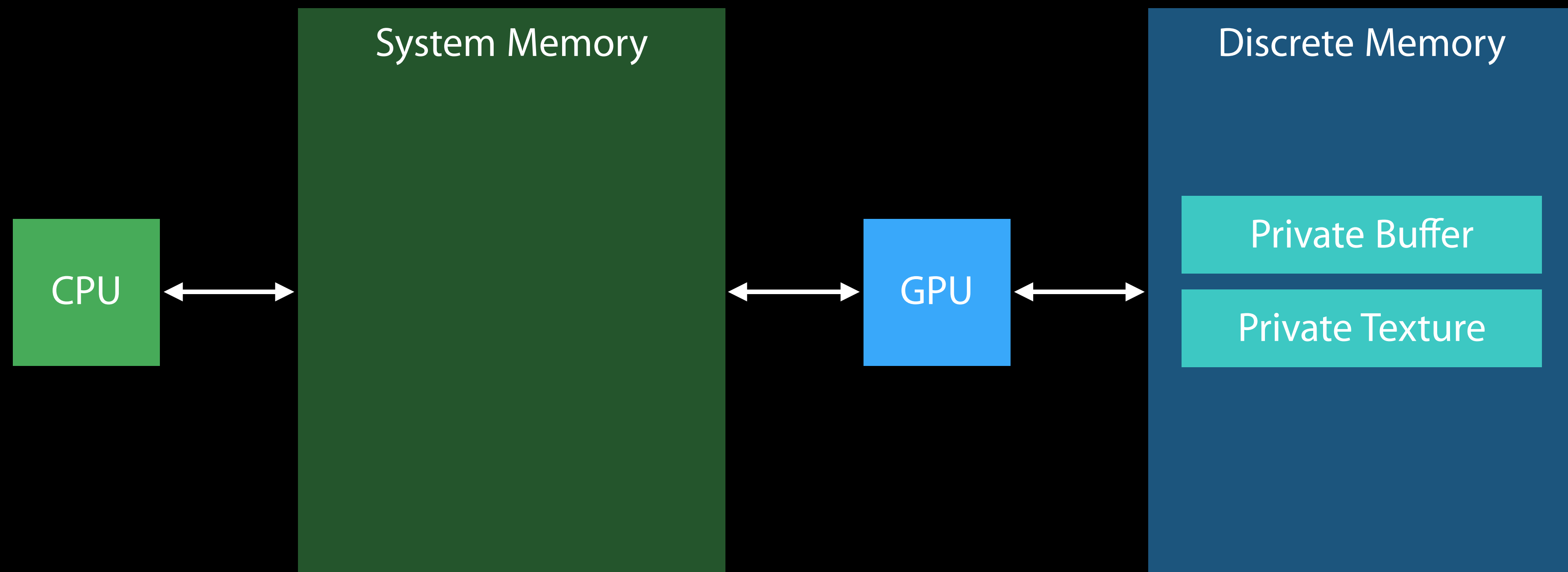
New with iOS 9 and OS X

Resources are only accessible to GPU—blit, render, compute

Metal can store data more optimally for the GPU



# Private Storage Mode with Discrete Memory



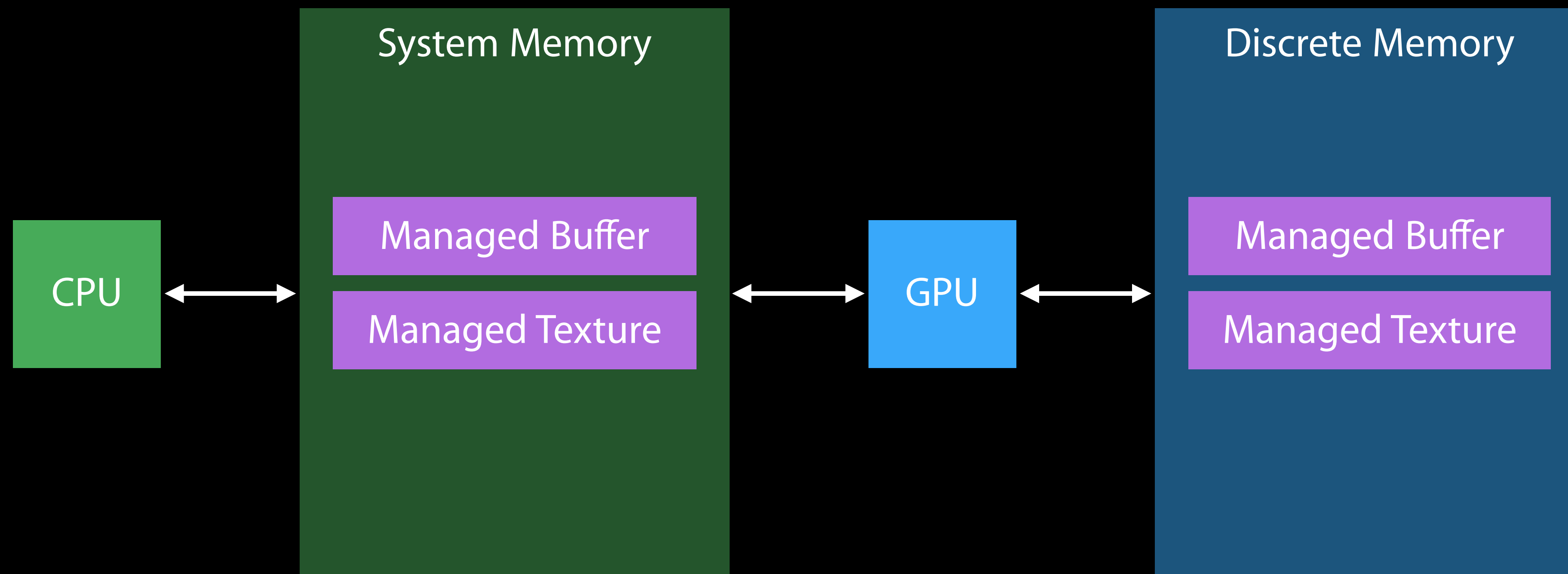


# Managed Storage Mode

New with OS X

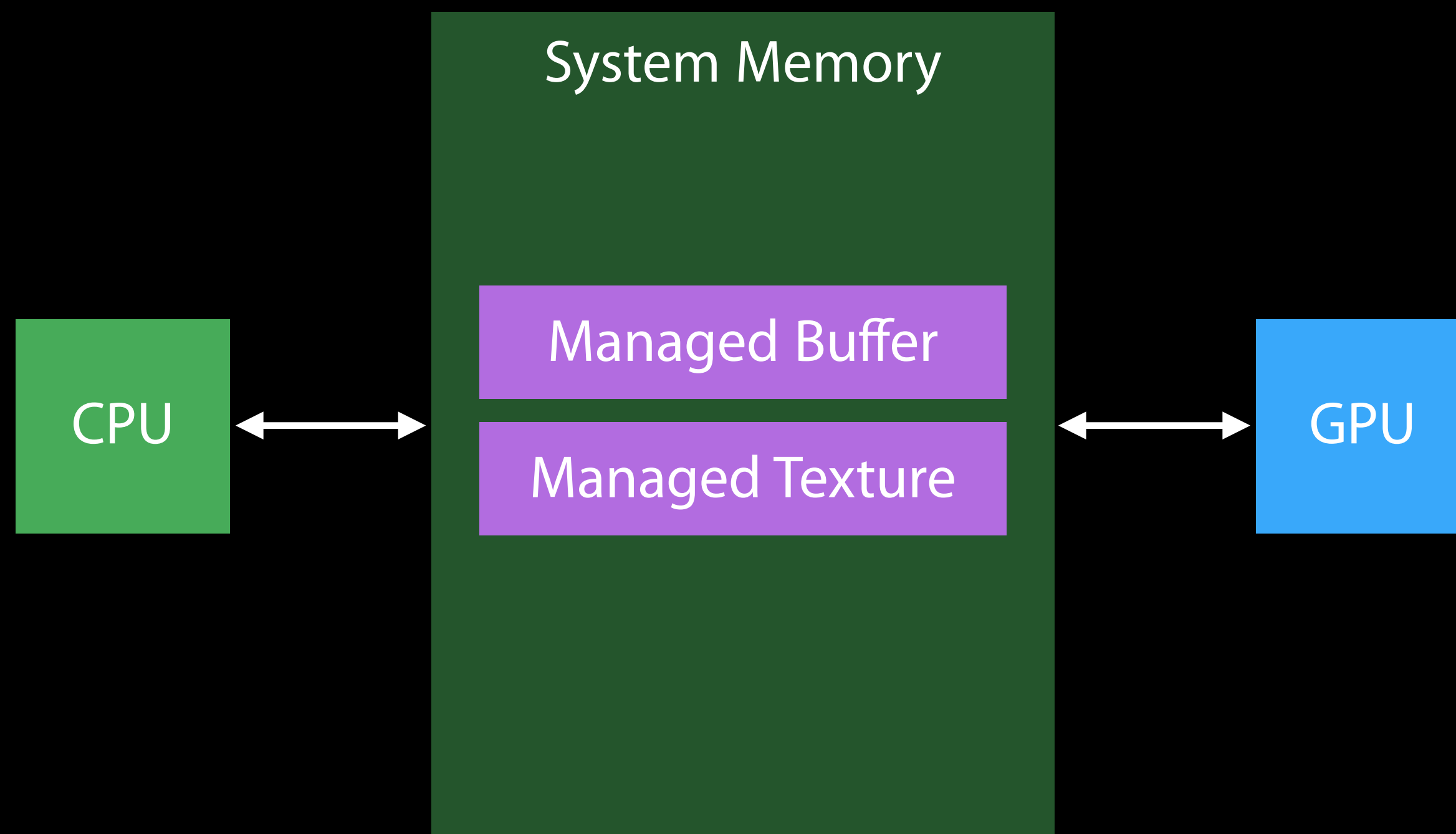
Metal manages where the resource resides

Performance of private memory with convenience of shared



# Managed Storage Mode

No extra copy for unified memory systems



# Managed Storage Mode

CPU modified data

App must notify Metal when modifying a resource with CPU

```
[myBuffer didModifyRange:...];
```

```
[myTexture replaceRegion:...];
```



# Managed Storage Mode

CPU read back

App must synchronize resource before CPU read

```
[blitCmdEncoder synchronizeResource:myBuffer];  
[cmdBuffer waitUntilCompleted]; // Or use completion handler  
contents = [myBuffer contents];
```

```
[blitCmdEncoder synchronizeResource:myTexture];  
[cmdBuffer waitUntilCompleted]; // Or use completion handler  
[myTexture getBytes:...];
```

# Shader Constant Update

## Review

```
id <MTLBuffer> constant_buffer = ...;
MyConstants* constant_ptr = constant_buffer.contents;

[renderpass setVertexBuffer: constant_buffer offset: 0 atIndex: 0];

for (i=0; i<draw_count; i++)
{
    constant_ptr[i] = // write constants directly into the buffer
    [renderpass setVertexBufferOffset:i*sizeof(MyConstants) atIndex:0];

    // draw
}
```

# Constant Updates on Discrete Systems

## Using Managed Buffers

Fast and easy shader constant uploads

```
id <MTLBuffer> constant_buffer = [device newBufferWithOptions:MTLResourceStorageModeManaged
                                                                    length:kMyConstantBufferSize];
MyConstants* constant_ptr = constant_buffer.contents;

[renderpass setVertexBuffer: constant_buffer offset: 0 atIndex: 0];

foreach i in draw_count
{
    constant_ptr[i] = // write constants directly into the buffer
    [renderpass setVertexBufferOffset:i*sizeof(MyConstants) atIndex:0];

    // draw
}

[constant_buffer didModifyRange:NSMakeRange(0, i*sizeof(MyConstants))];
```



# Constant Updates on Discrete Systems

## Using Managed Buffers

Fast and easy shader constant uploads

```
id <MTLBuffer> constant_buffer = [device newBufferWithOptions:MTLResourceStorageModeManaged  
                                                                    length:kMyConstantBufferSize];
```

```
MyConstants* constant_ptr = constant_buffer.contents;
```

```
[renderpass setVertexBuffer: constant_buffer offset: 0 atIndex: 0];
```

```
foreach i in draw_count
```

```
{
```

```
    constant_ptr[i] = // write constants directly into the buffer
```

```
    [renderpass setVertexBufferOffset:i*sizeof(MyConstants) atIndex:0];
```

```
    // draw
```

```
}
```

```
[constant_buffer didModifyRange:NSMakeRange(0, i*sizeof(MyConstants))];
```

# Constant Updates on Discrete Systems

## Using Managed Buffers

Fast and easy shader constant uploads

```
id <MTLBuffer> constant_buffer = [device newBufferWithOptions:MTLResourceStorageModeManaged  
                                                                    length:kMyConstantBufferSize];
```

```
MyConstants* constant_ptr = constant_buffer.contents;
```

```
[renderpass setVertexBuffer: constant_buffer offset: 0 atIndex: 0];
```

```
foreach i in draw_count
```

```
{  
    constant_ptr[i] = // write constants directly into the buffer  
    [renderpass setVertexBufferOffset:i*sizeof(MyConstants) atIndex:0];
```

```
    // draw
```

```
}  
[constant_buffer didModifyRange:NSMakeRange(0, i*sizeof(MyConstants))];
```

# Metal Managed Memory Model

## Default storage modes

Buffers are shared

Default texture storage mode depends on platform

- iOS default is shared
- OS X default is managed



# Customizing Storage Modes

## Private textures

Use private for GPU-only resources

```
fbTextureDesc = [MTLTextureDescriptor texture2DDescriptorWithPixelFormat:myColorFormat
                                                    width:myWidth
                                                    height:myHeight
                                                    mipmapped:NO];

fbTextureDesc.storageMode = MTLStorageModePrivate;

fbTexture = [device newTextureWithDescriptor:fbTextureDesc];
```

# Customizing Storage Modes

## Private textures

Use private for GPU-only resources

```
fbTextureDesc = [MTLTextureDescriptor texture2DDescriptorWithPixelFormat:myColorFormat
                                                    width:myWidth
                                                    height:myHeight
                                                    mipmapped:NO];
```

```
fbTextureDesc.storageMode = MTLStorageModePrivate;
```

```
fbTexture = [device newTextureWithDescriptor:fbTextureDesc];
```

# Device Selection

On multi-GPU systems

Use `MTLCreateSystemDefaultDevice`

- Picks the device connected to the main display
- Activates the discrete GPU on systems with automatic graphics switching



# Device Selection

## Selecting the Auxiliary GPU on a Mac Pro

New `MTLCopyAllDevices` API to enumerate all Metal capable devices

- 'headless' property identifies auxiliary GPU

```
id <MTLDevice> aux_gpu = nil;
for (id <MTLDevice> device in MTLCopyAllDevices())
{
    if ([device isHeadless]) {
        aux_gpu = device;
        break;
    }
}
```

# Device Selection

## Selecting the Auxiliary GPU on a Mac Pro

New `MTLCopyAllDevices` API to enumerate all Metal capable devices

- 'headless' property identifies auxiliary GPU

```
id <MTLDevice> aux_gpu = nil;
for (id <MTLDevice> device in MTLCopyAllDevices())
{
    if ([device isHeadless]) {
        aux_gpu = device;
        break;
    }
}
```

# Device Selection

## Selecting the Auxiliary GPU on a Mac Pro

New `MTLCopyAllDevices` API to enumerate all Metal capable devices

- 'headless' property identifies auxiliary GPU

```
id <MTLDevice> aux_gpu = nil;
for (id <MTLDevice> device in MTLCopyAllDevices())
{
    if ([device isHeadless]) {
        aux_gpu = device;
        break;
    }
}
```

# Device Selection

Selecting the 'best' device in a dual-GPU MacBook Pro

Ideal for applications that are not full-screen games and want to optimize for power



# Device Selection

Selecting the 'best' device in a dual-GPU MacBook Pro

Ideal for applications that are not full-screen games and want to optimize for power

Register for a GPU 'switch' notification

- `NSNotificationGlobalFrameDidChangeNotification`

# Device Selection

## Selecting the 'best' device in a dual-GPU MacBook Pro

Ideal for applications that are not full-screen games and want to optimize for power

Register for a GPU 'switch' notification

- `NSNotificationGlobalFrameDidChangeNotification`

Use the CoreGraphics convenience API to query the current active device

- Pass in the current display your view is on

```
#include <CoreGraphics/CGDirectDisplayMetal.h>
```

```
id <MTLDevice> device = = CGDirectDisplayCopyCurrentMetalDevice(display);
```

# Layered Rendering

Rasterize to multiple layers of a texture

- Slices of a 2D array texture
- Plane of a 3D texture
- Face of a cube texture

# Layered Rendering

Rasterize to multiple layers of a texture

- Slices of a 2D array texture
- Plane of a 3D texture
- Face of a cube texture

Output the target layer from a vertex shader

```
struct VSOut {  
    float4 position [[position]];  
    ushort layer [[render_target_array_index]];  
}
```

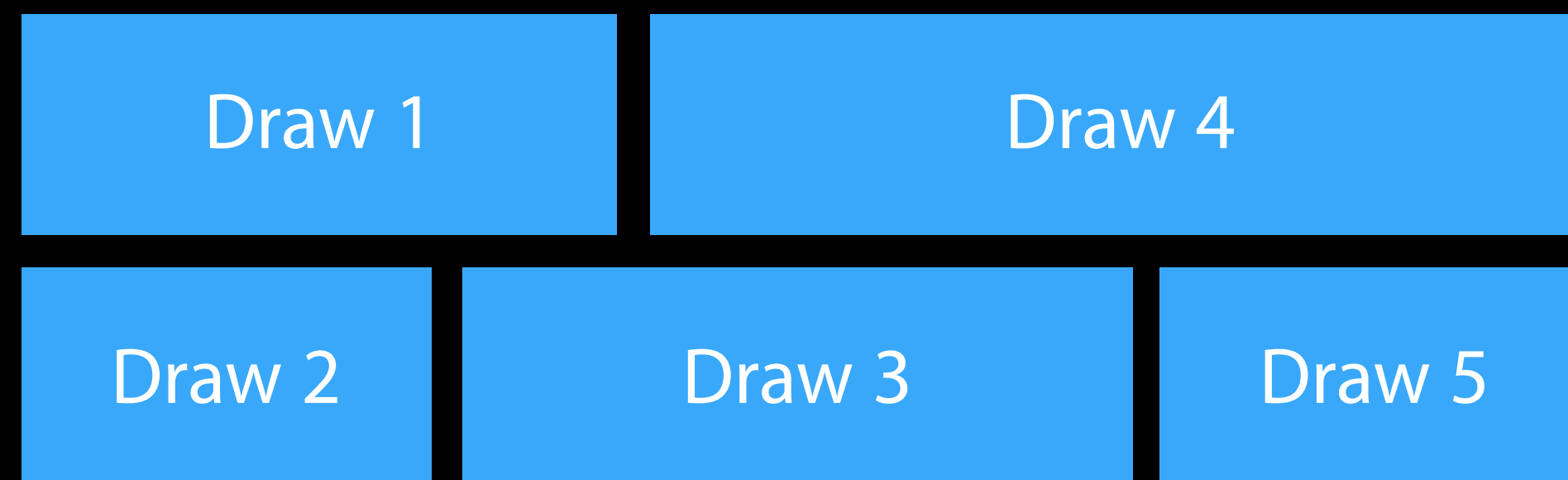


# Texture Barriers

GPUs overlap execution of many draw calls

Output of one draw cannot be safely read by a later draw

New API to insert barriers between draw calls

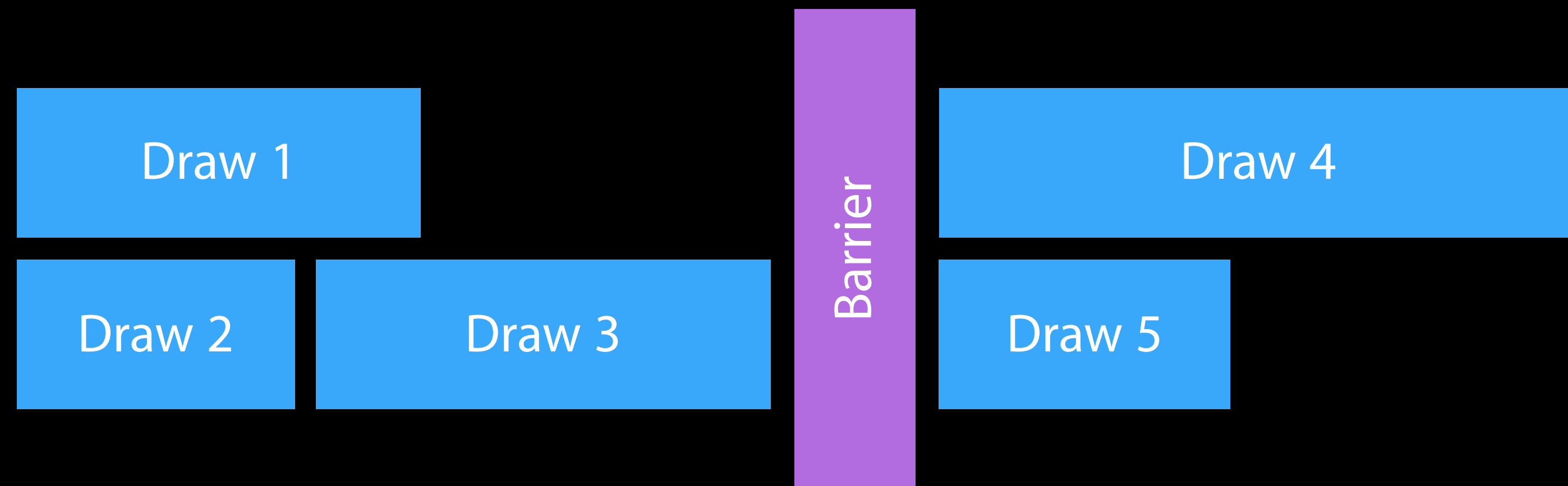


# Texture Barriers

GPUs overlap execution of many draw calls

Output of one draw cannot be safely read by a later draw

New API to insert barriers between draw calls



# Texture Barriers

## Example

```
// start render pass, drawing to Texture A  
[renderPass draw...];
```

```
[renderPass setFragmentTexture:textureA atIndex:0];  
[renderPass draw...];
```

# Texture Barriers

## Example

```
// start render pass, drawing to Texture A  
[renderPass draw...];  
  
[renderPass textureBarrier];  
  
[renderPass setFragmentTexture:textureA atIndex:0];  
[renderPass draw...];
```



# New Texture Features

	iOS GPUFamily1	iOS GPUFamily2	OS X GPUFamily1
Max Number of Textures per Shader Stage	31	31	128
Max Texture Size	8k	8k	16k
Max Render Target Count	4	8	8
MSAA	2x, 4x	2x, 4x	4x, 8x
Cube Array Support	-	-	Yes
Compute Pixel Writes	Int32, Float32	Int32, Float32	Int32, Float32, packed

# New Texture Features

## Texture usage

New property in the texture descriptor to declare how a texture will be used

Allows the Metal implementation to optimize for that usage

`MTLTextureUsageUnknown`

`MTLTextureUsageShaderRead`

`MTLTextureUsageShaderWrite`

`MTLTextureUsageRenderTarget`

`MTLTextureUsageBlit`

# New Texture Features

## Texture usage

New property in the texture descriptor to declare how a texture will be used

Allows the Metal implementation to optimize for that usage

MTLTextureUsageUnknown

MTLTextureUsageShaderRead

MTLTextureUsageShaderWrite

MTLTextureUsageRenderTarget

MTLTextureUsageBlit

# New Texture Features

## Texture usage

New property in the texture descriptor to declare how a texture will be used

Allows the Metal implementation to optimize for that usage

**MTLTextureUsageUnknown**

MTLTextureUsageShaderRead

MTLTextureUsageShaderWrite

MTLTextureUsageRenderTarget

MTLTextureUsageBlit



# New Texture Features

## Depth/stencil textures

Mac GPUs only support combined depth and stencil formats

- Depth32Float\_stencil8
  - Supported on all Metal Devices
- Depth24Unorm\_stencil8
  - If available and meets your precision requirements

# New Texture Features

## iOS texture compression formats

Format	Bits Per Pixel	Support	Properties
PVRTC	2, 4	All iOS devices	RGB content Widest support
ETC2	4 - RGB 8 - RGBA	All Metal devices	RGB content Good alpha support
EAC	4 - One channel 8 - Two channels	All Metal devices	Height/Bump Maps Normal Maps Alpha Masks
ASTC	0.9 - 8	iOS GPUFamily2	Highest quality at all sizes Many size vs. quality options Slowest encoding

# New Texture Features

## iOS texture compression formats

Format	Bits Per Pixel	Support	Properties
PVRTC	2, 4	All iOS devices	RGB content Widest support
ETC2	4 - RGB 8 - RGBA	All Metal devices	RGB content Good alpha support
EAC	4 - One channel 8 - Two channels	All Metal devices	Height/Bump Maps Normal Maps Alpha Masks
ASTC	0.9 - 8	iOS GPUFamily2	Highest quality at all sizes Many size vs. quality options Slowest encoding

# New Texture Features

ASTC format



# New Texture Features

ASTC format

Very high-quality compression

# New Texture Features

## ASTC format

Very high-quality compression

Great for a broad range of usages

- Photographic content
- Height maps
- Normal maps
- Sprites

# New Texture Features

## ASTC format

Very high-quality compression

Great for a broad range of usages

- Photographic content
- Height maps
- Normal maps
- Sprites

Finer grained control of size vs. quality

- 1-8 bits per pixel

# New Texture Features

## ASTC format

Very high-quality compression

Great for a broad range of usages

- Photographic content
- Height maps
- Normal maps
- Sprites

Finer grained control of size vs. quality

- 1-8 bits per pixel

New in GPUFamily2



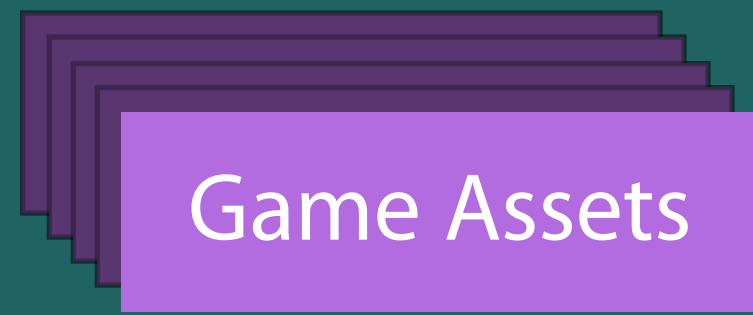
# New Texture Features

## OS X texture compression formats

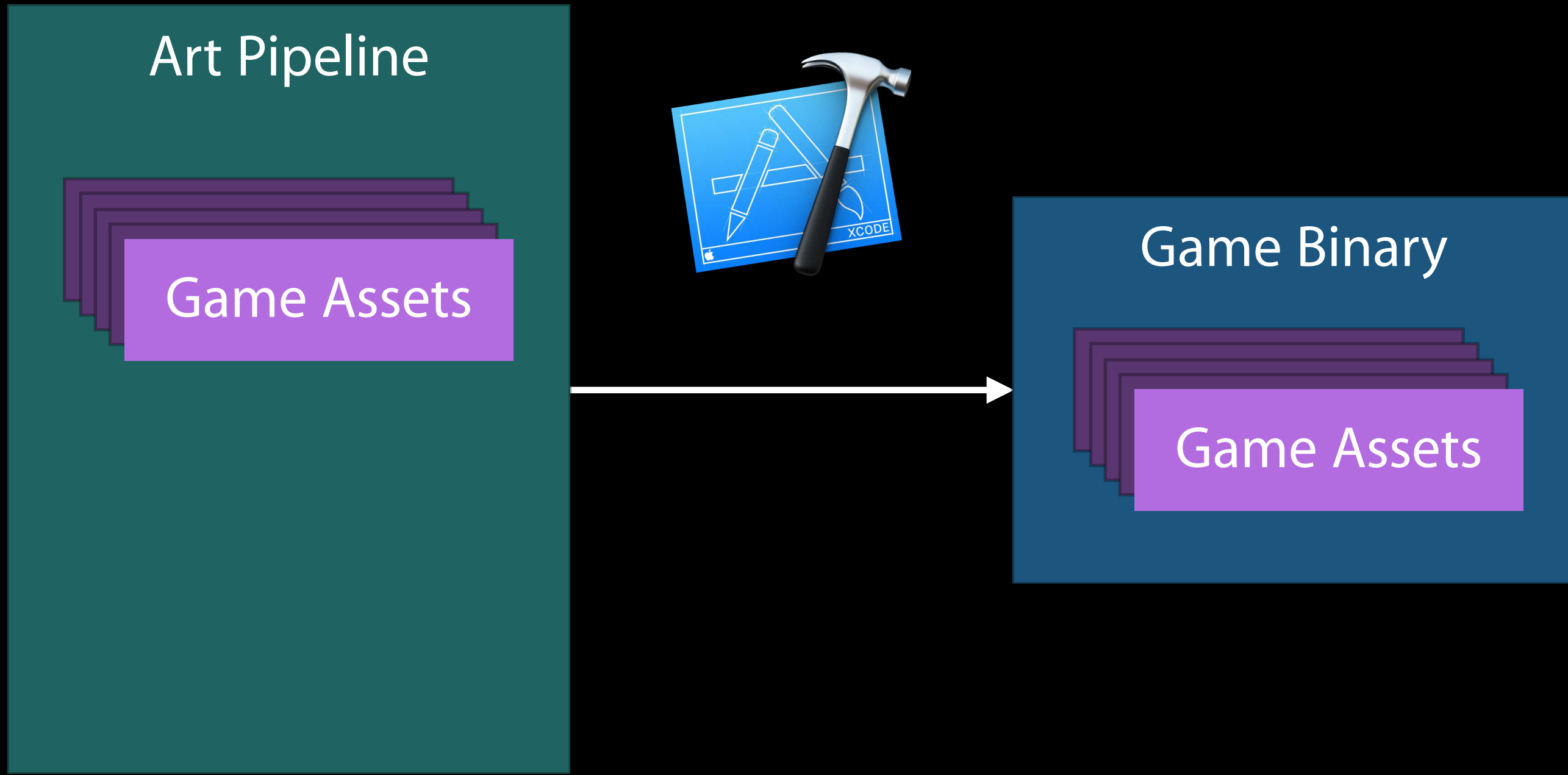
Format	Bits Per Pixel	Also Known As	Properties
BC1	4	S3TC, DXT1	RGB content Very fast encoding
BC2, BC3	8	S3TC, DXT3, DXT5	RGBA content Very fast encoding
BC4, BC5	4 - One channel 8 - Two channels	RGTC	Height/Bump Maps Normal Maps Alpha Masks
BC6, BC7	8	BPTC	RGBA content Slowest encoding

# Metal and App Thinning

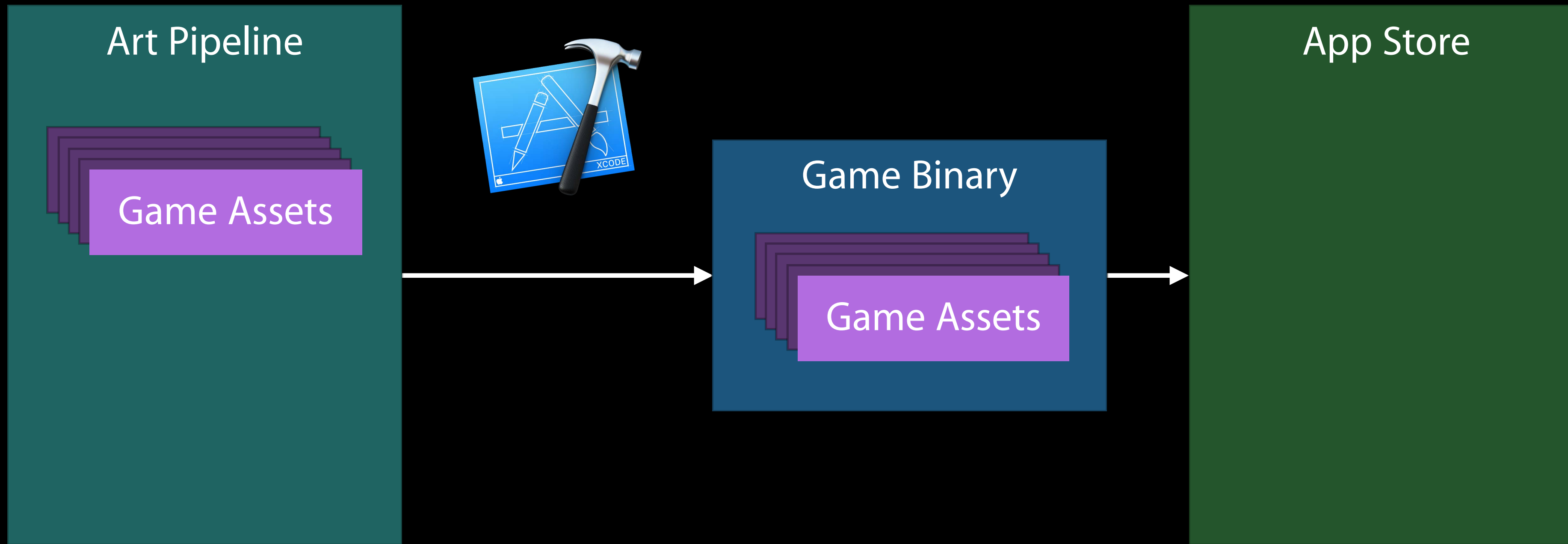
Art Pipeline

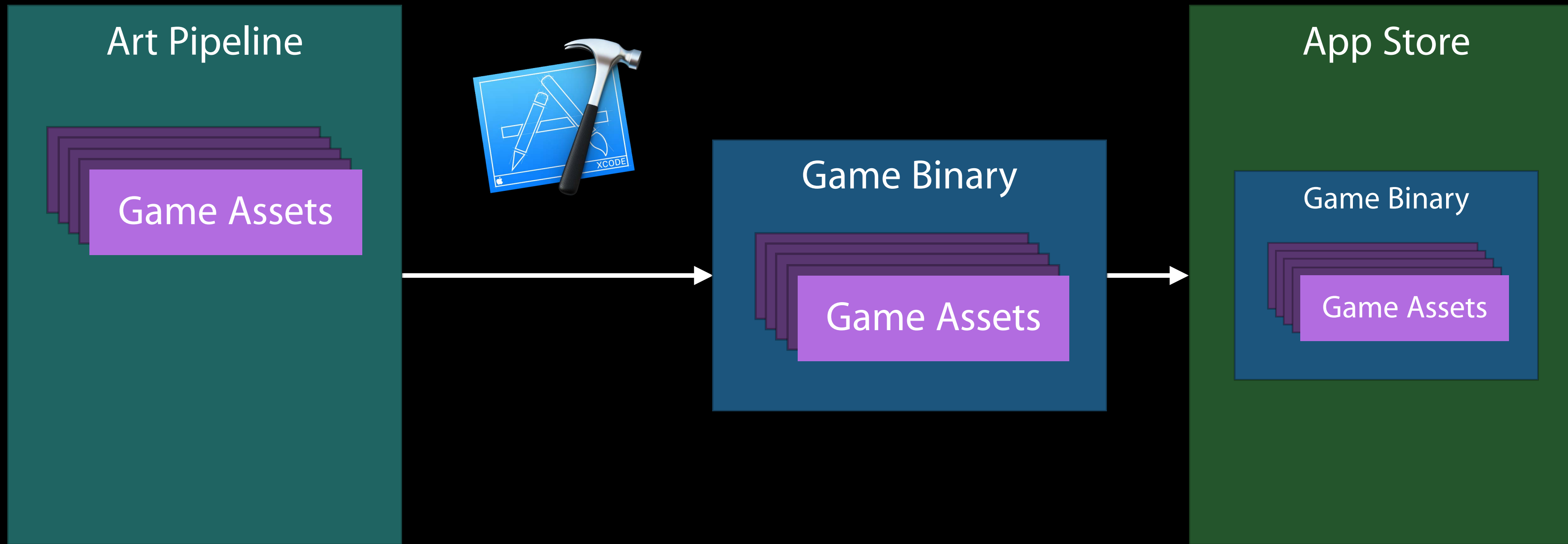


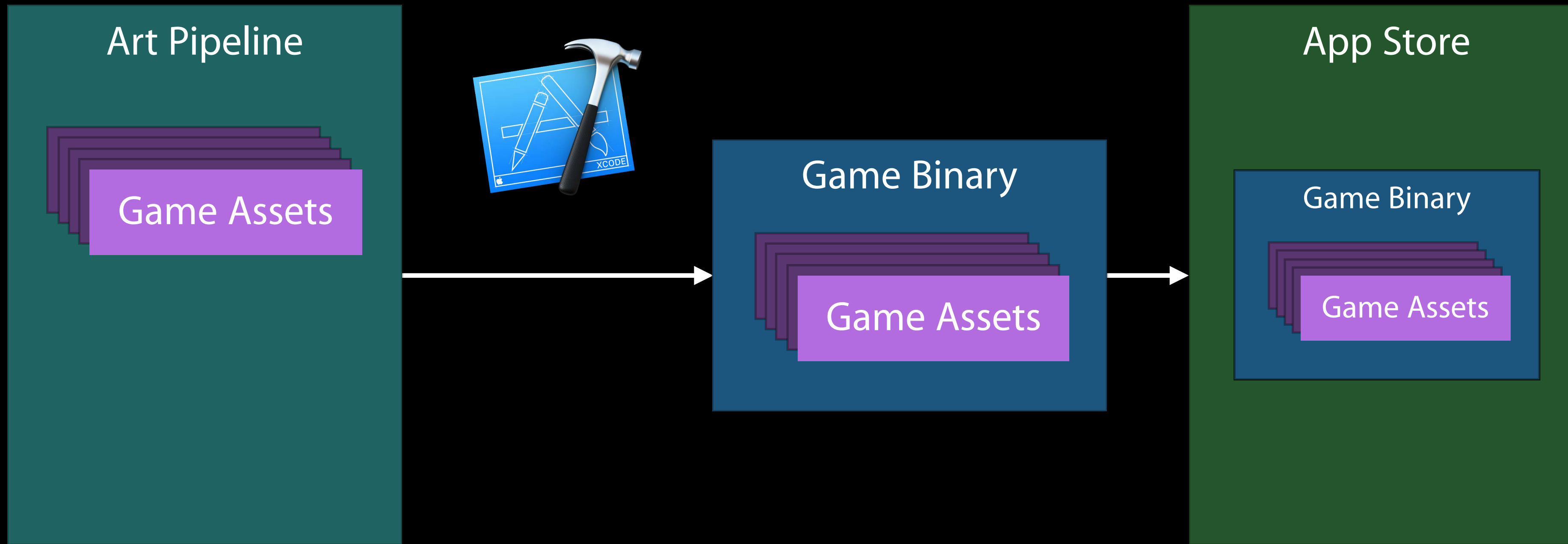
Game Assets









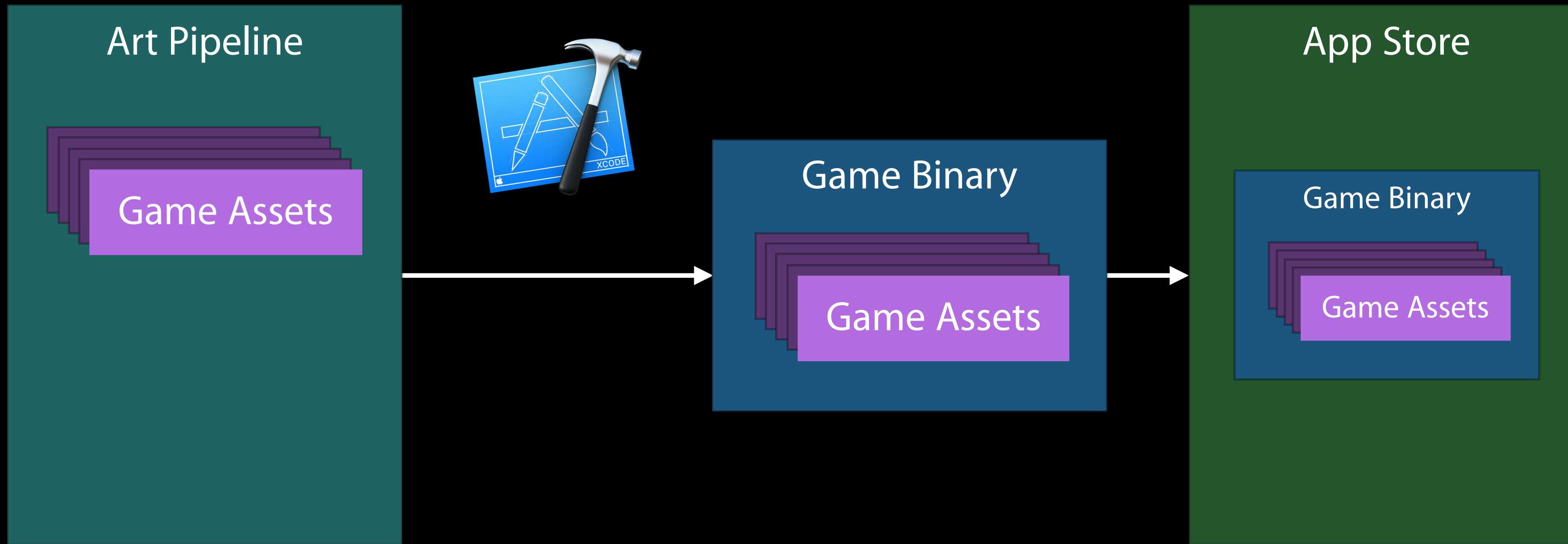


Legacy

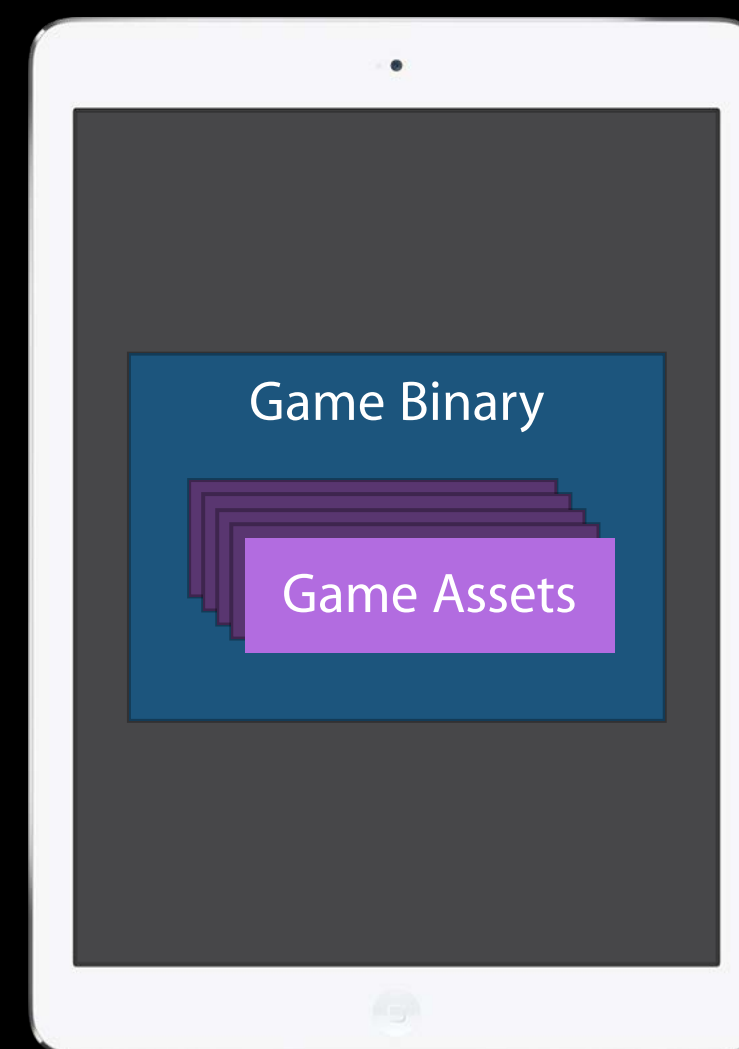


Metal Capable

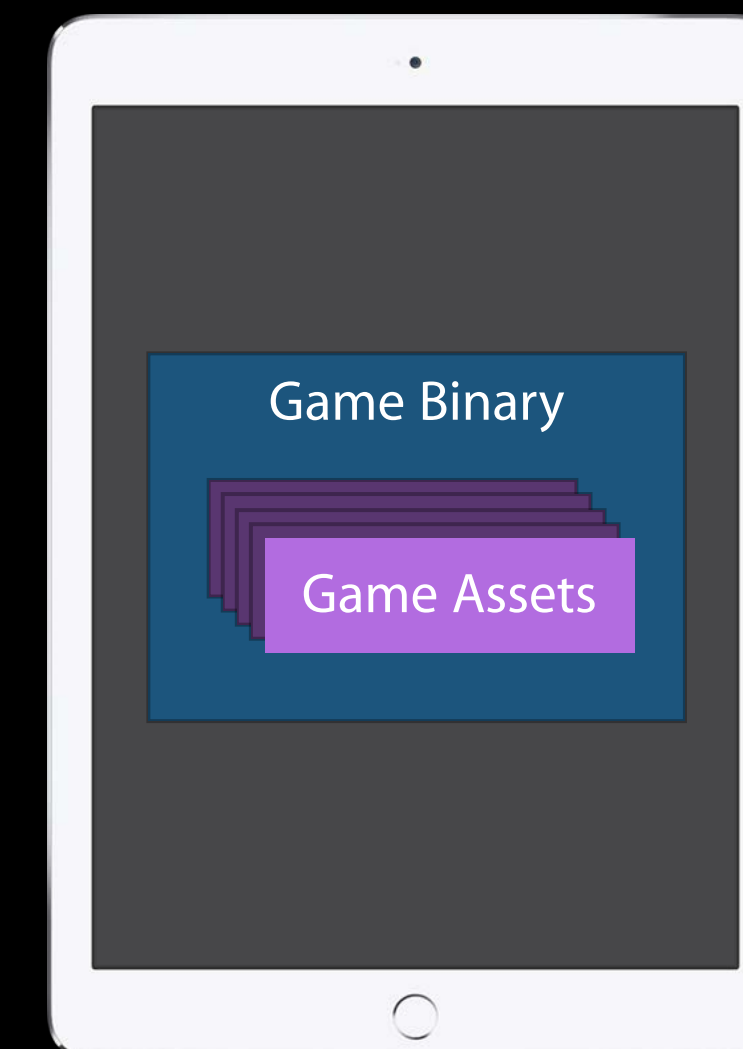




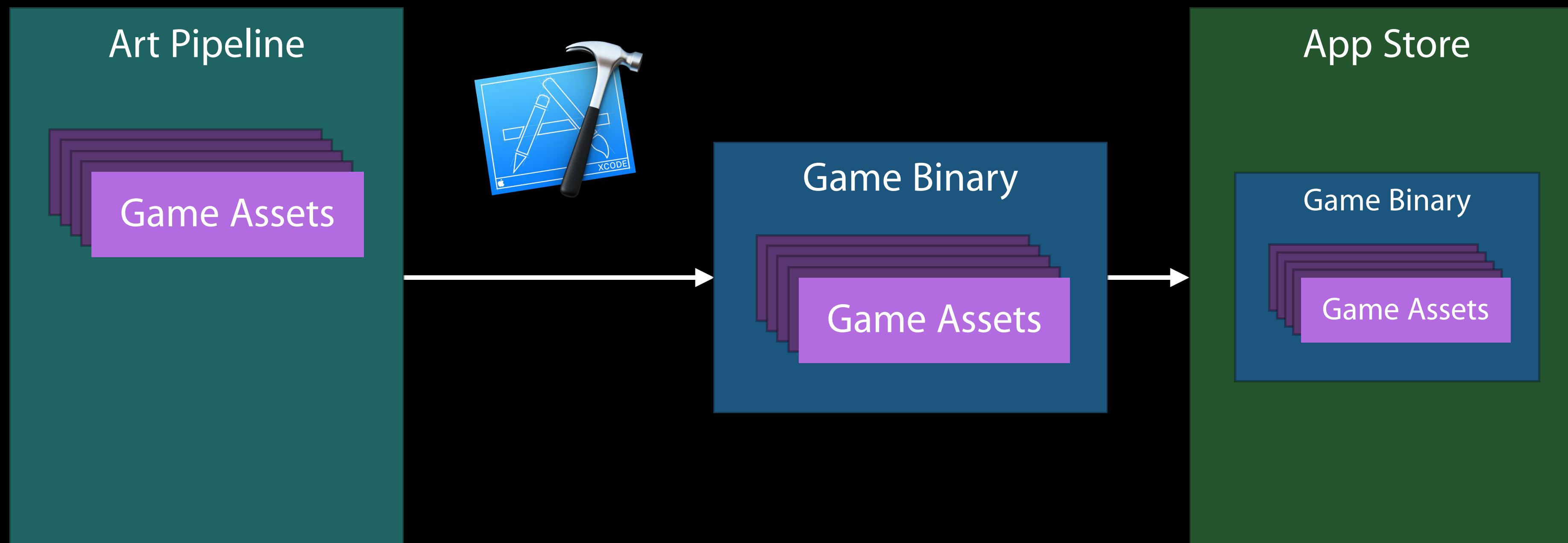
Legacy



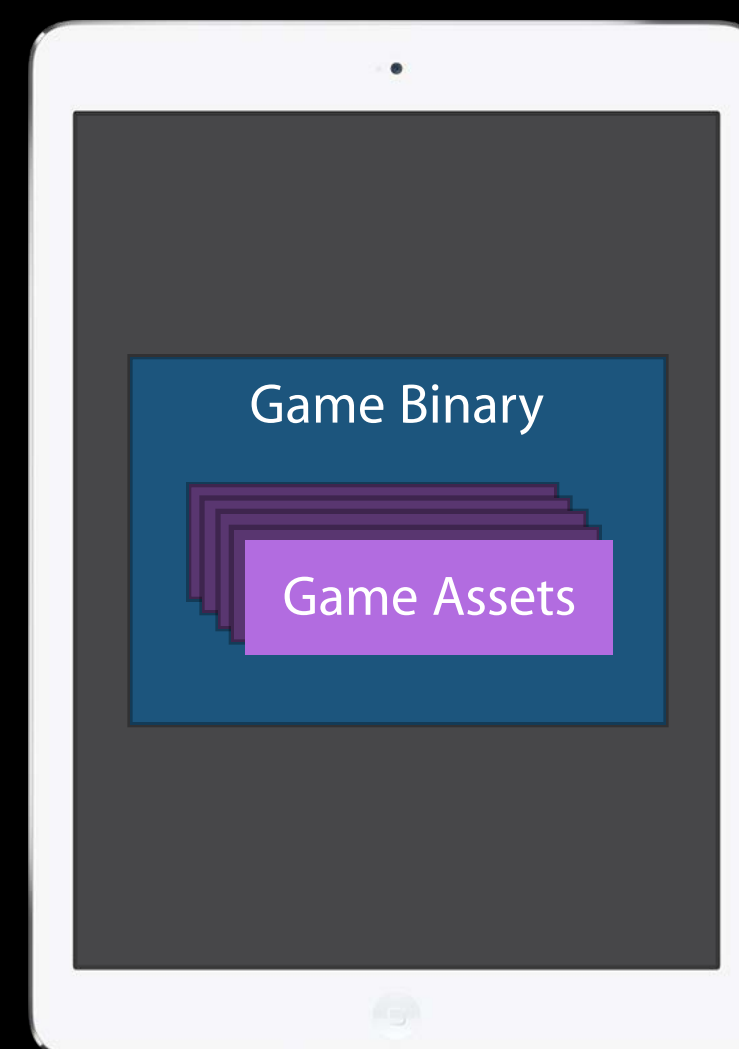
Metal Capable



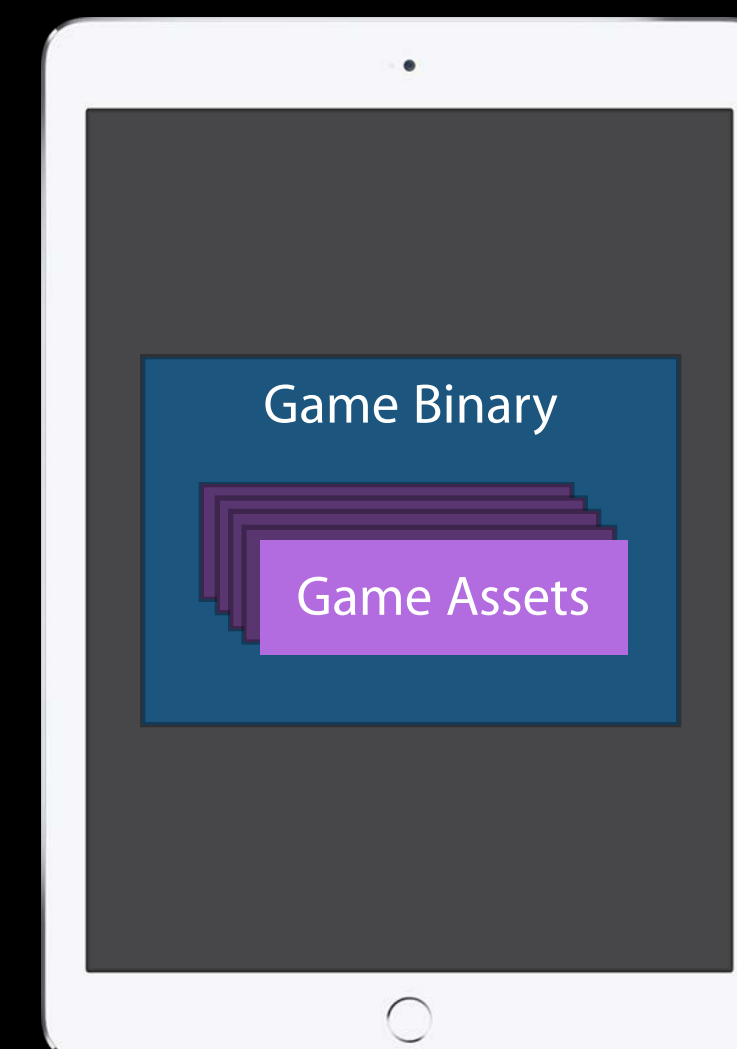


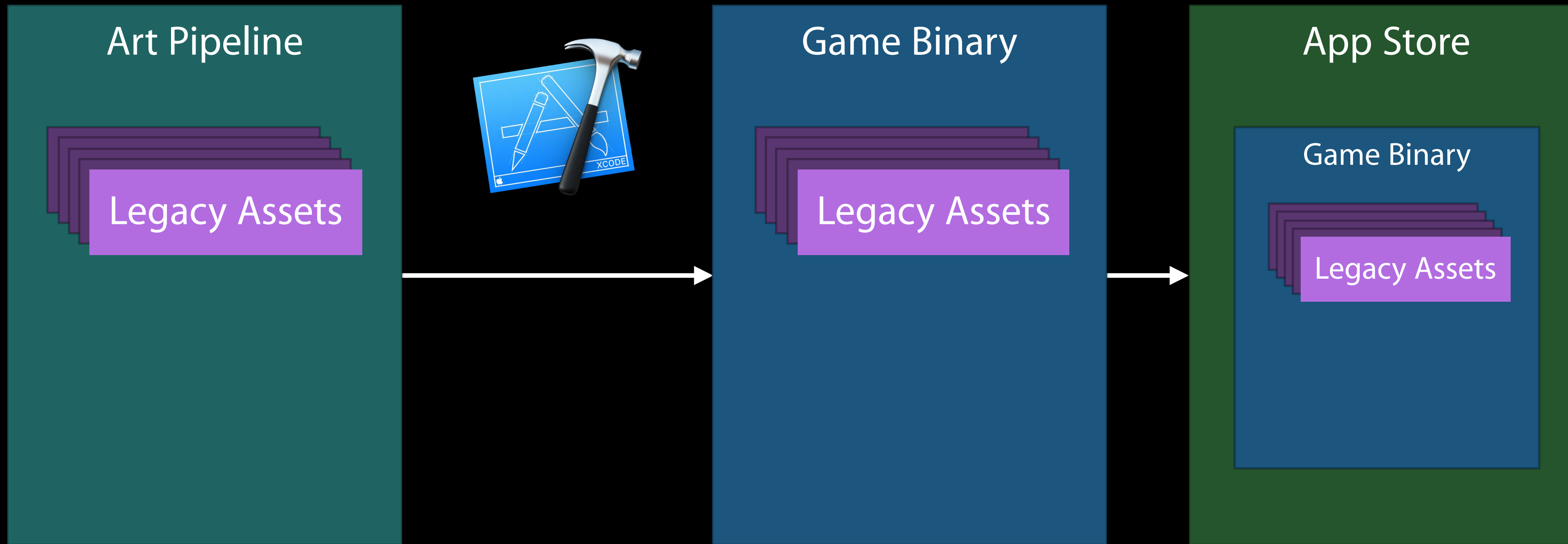


Legacy



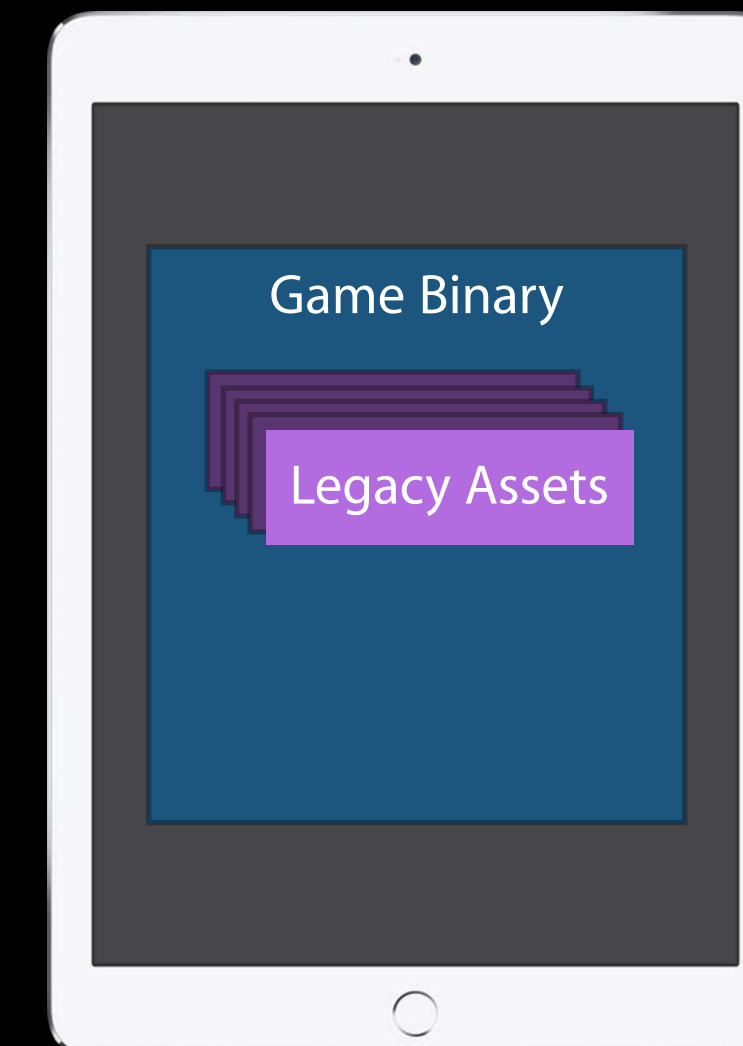
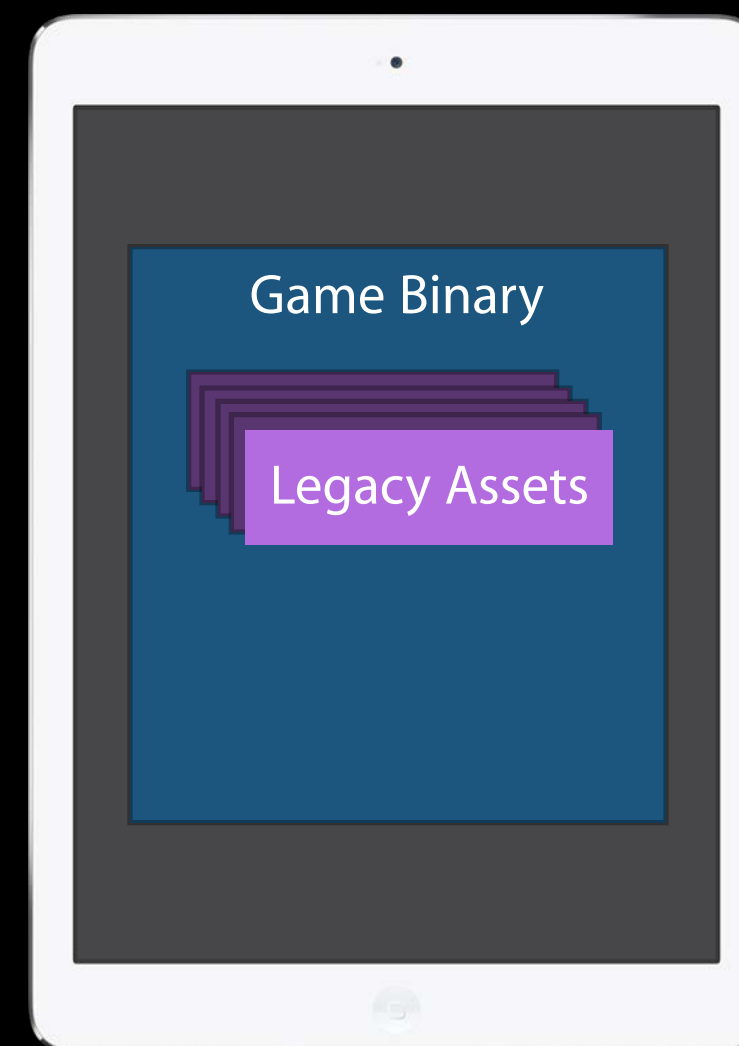
Metal Capable

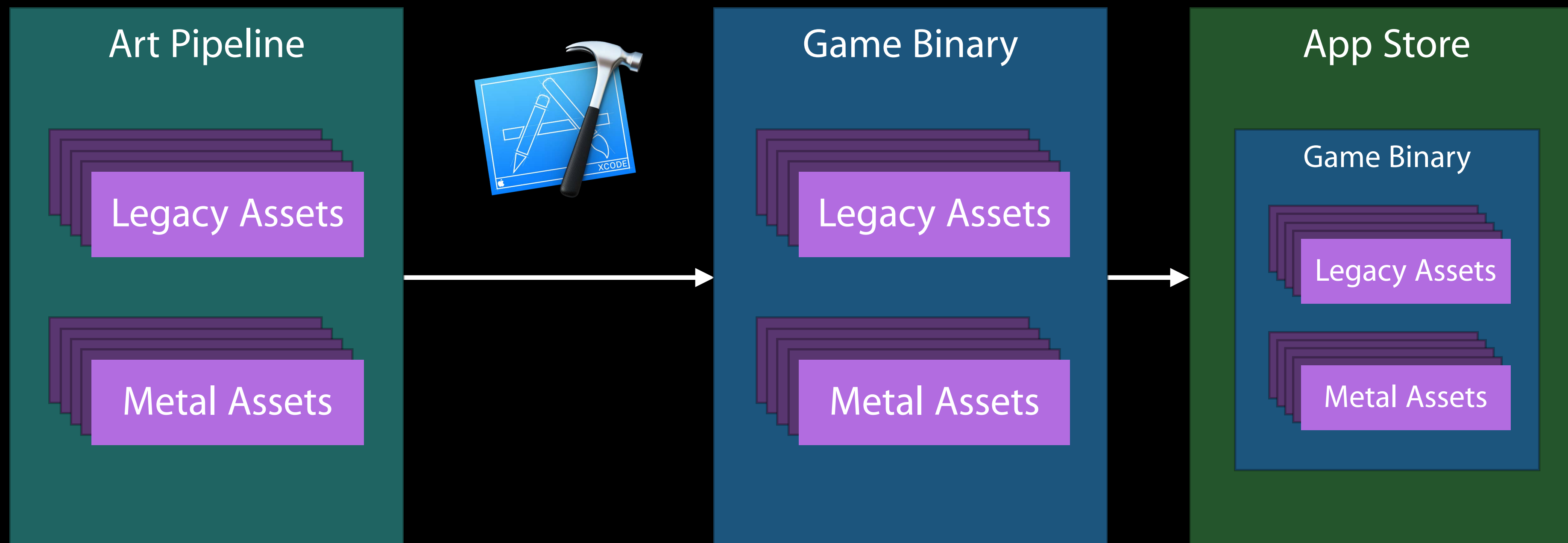




Legacy

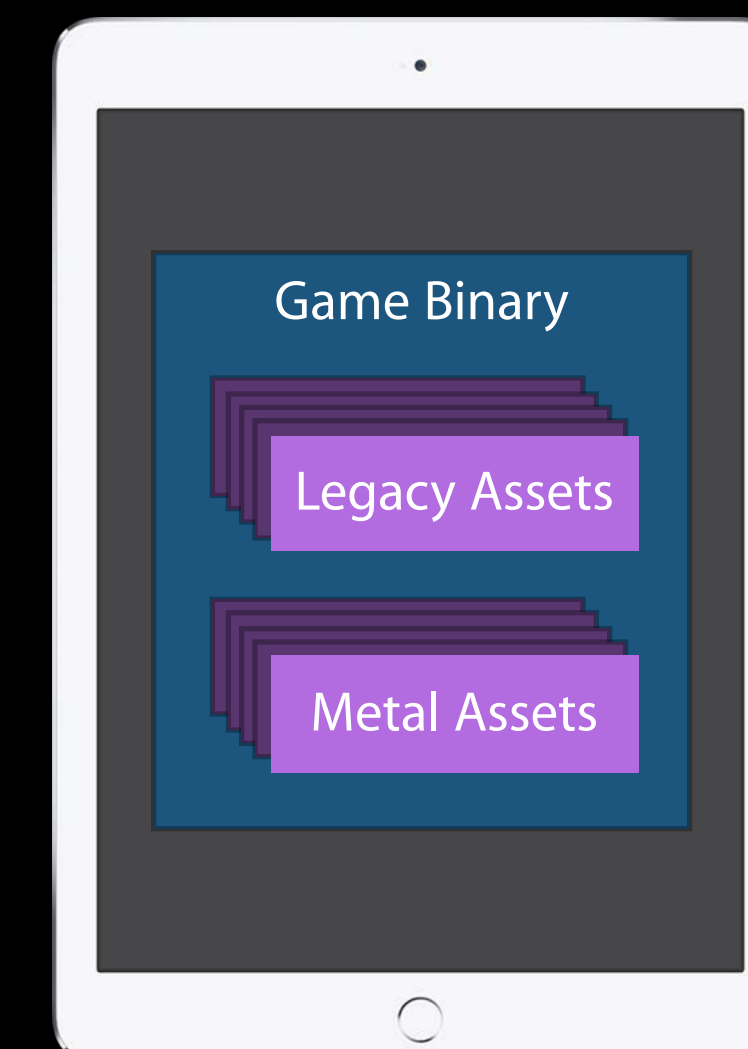
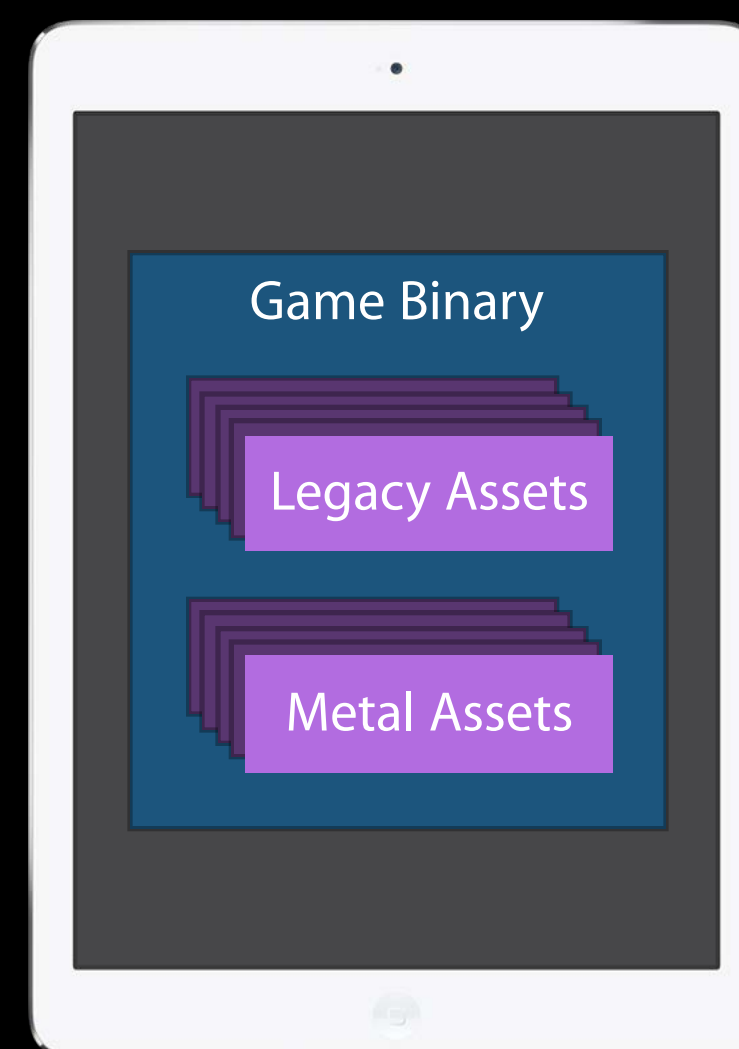
Metal Capable

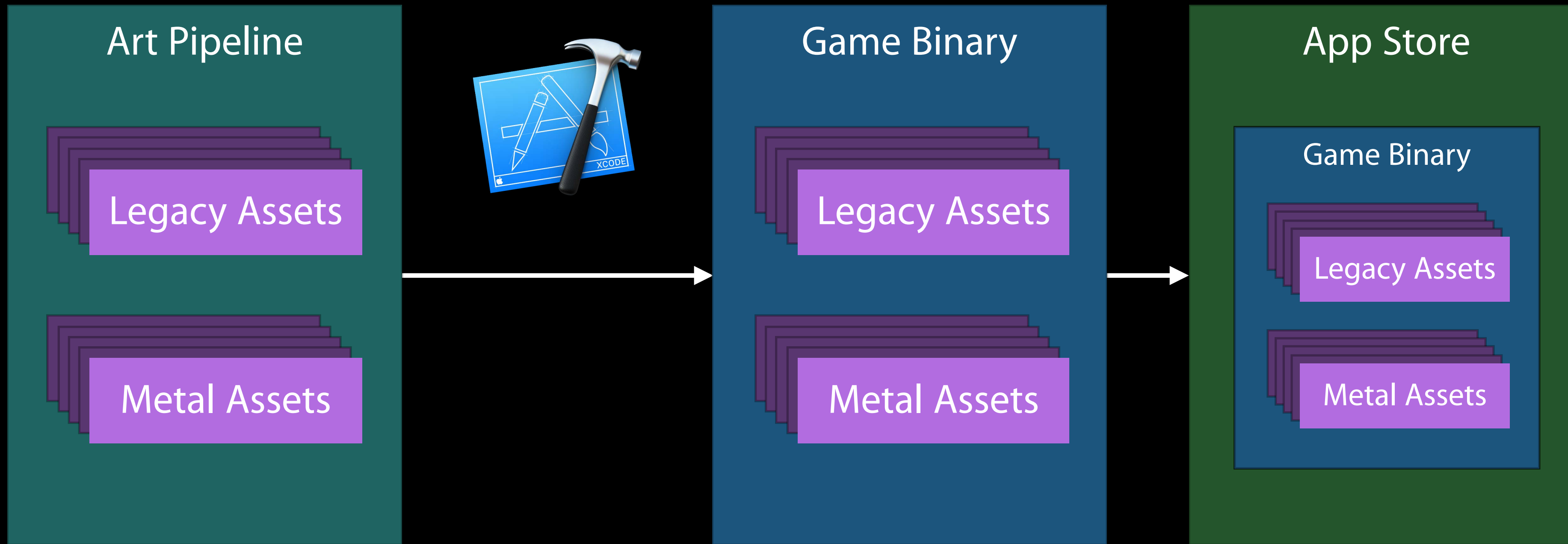




Legacy

Metal Capable





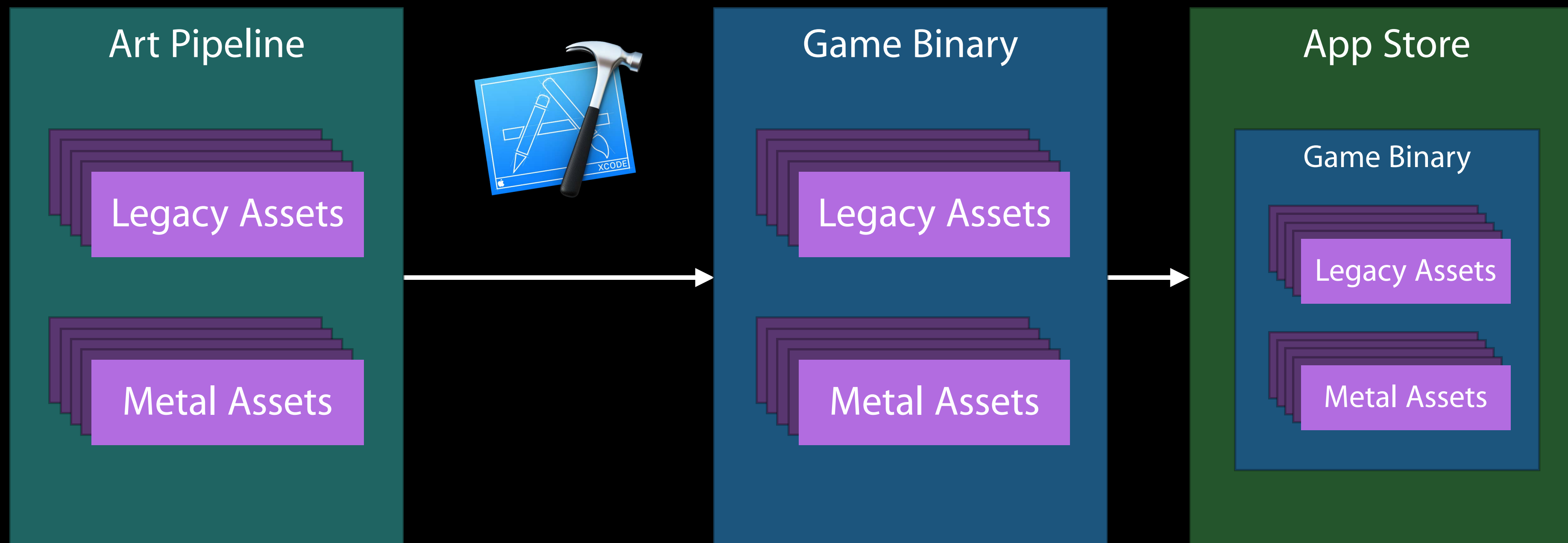
Legacy



Metal Capable

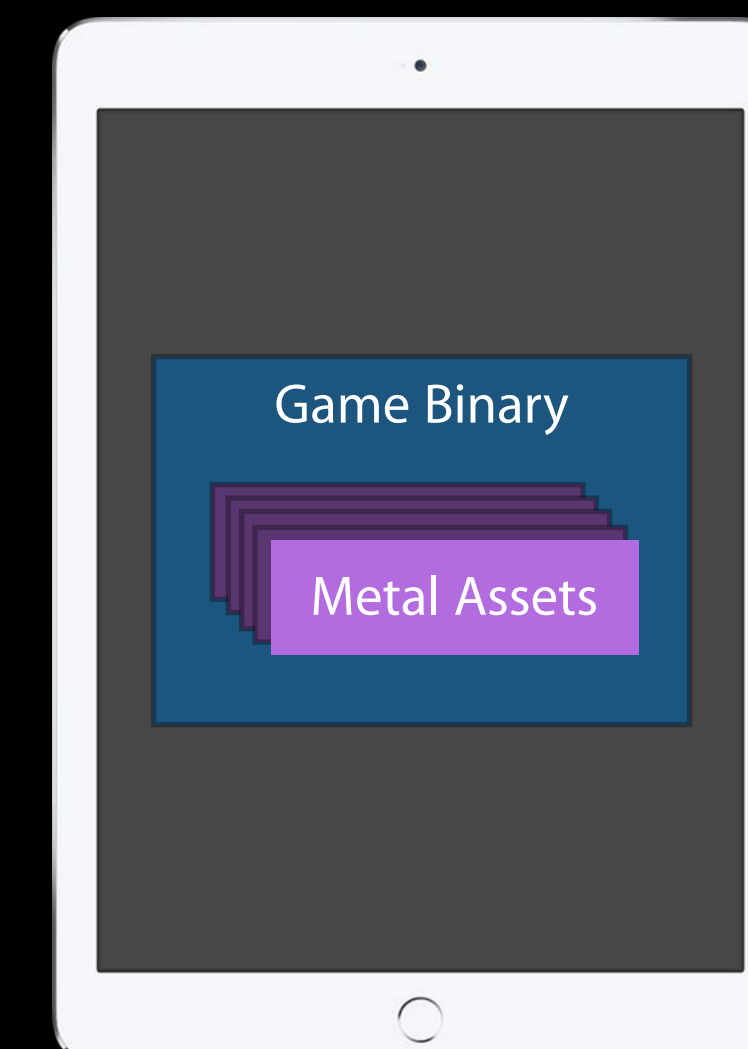
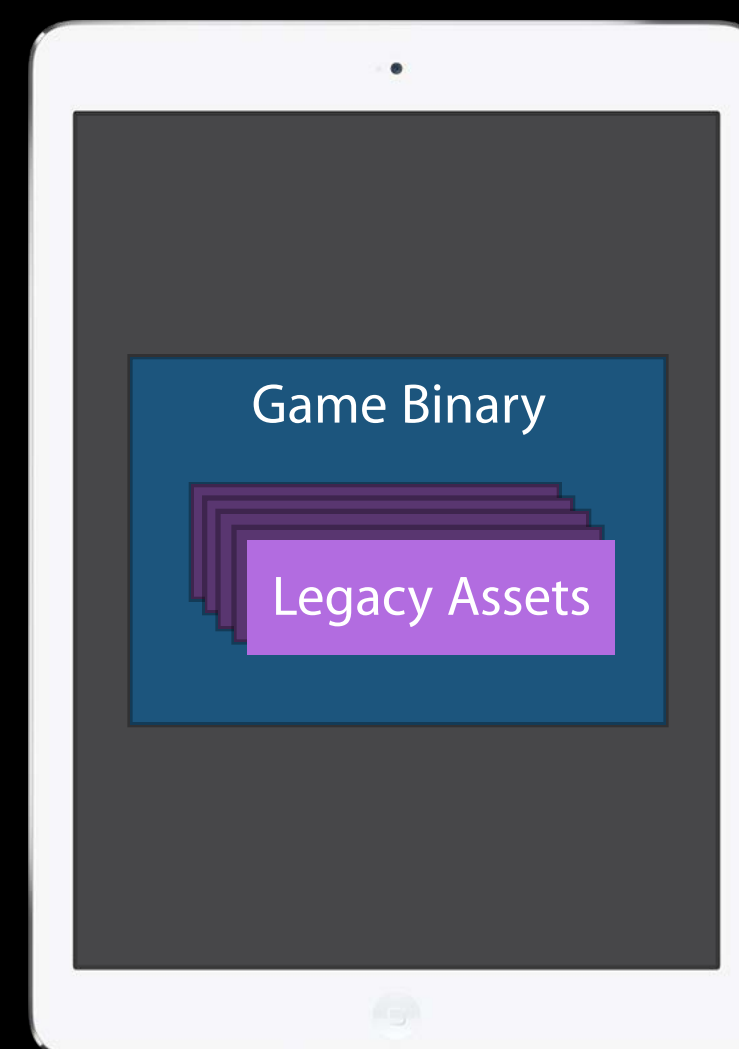


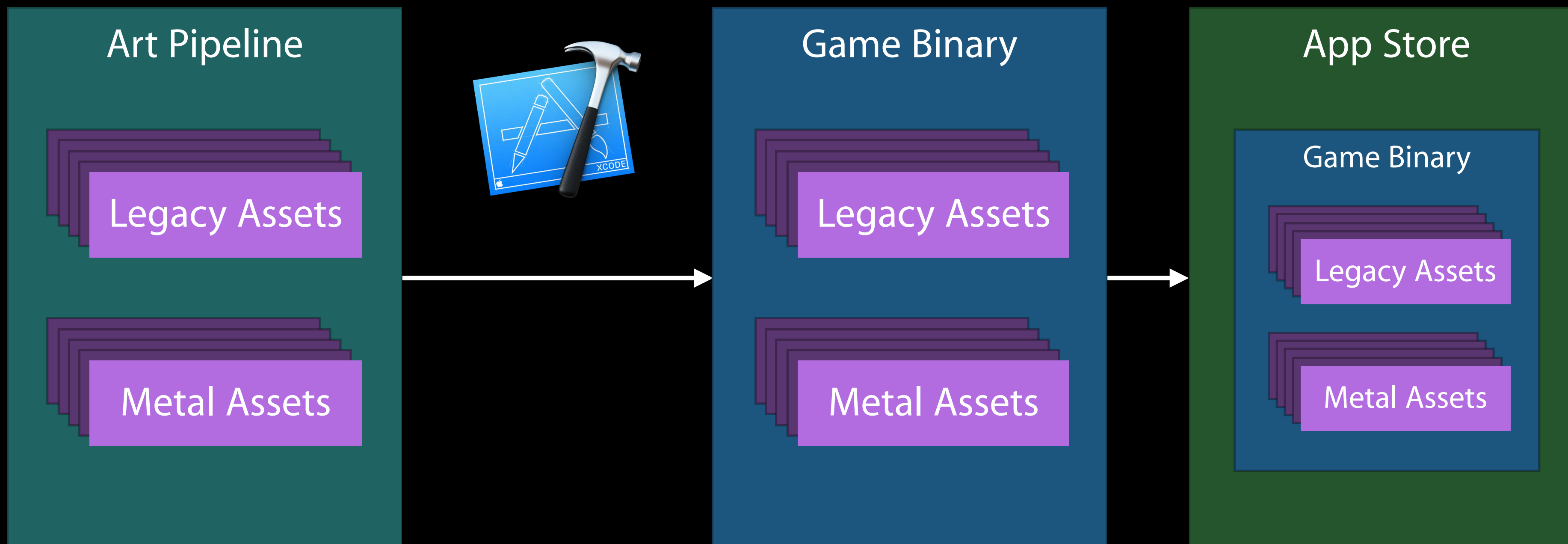




Legacy

Metal Capable

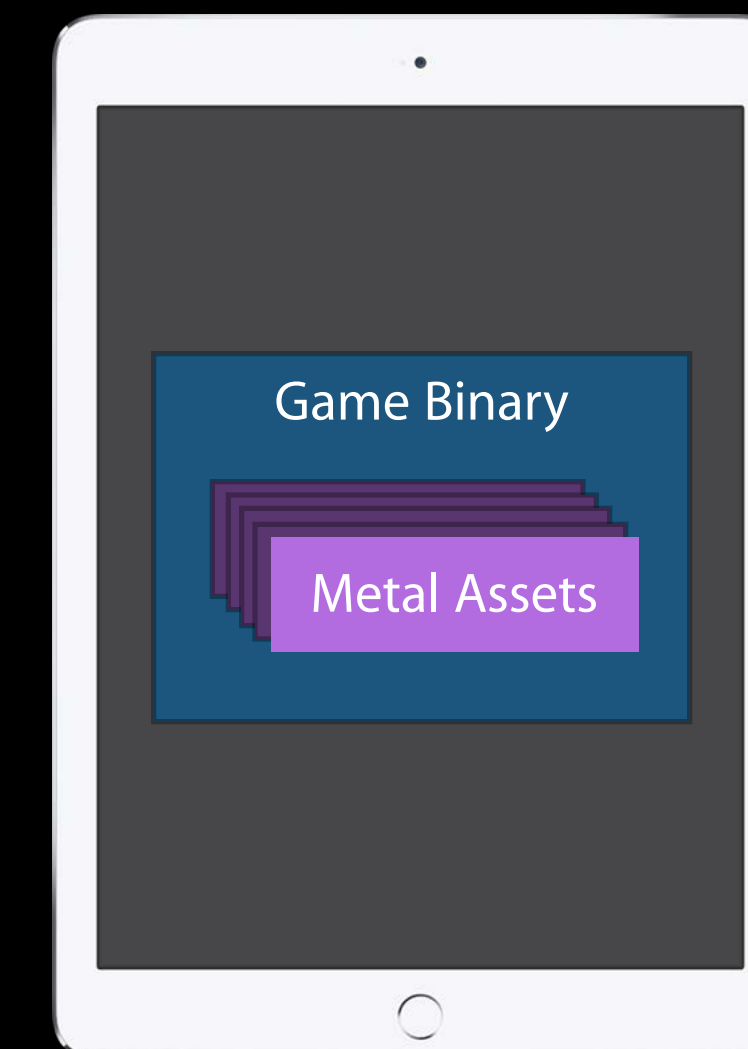
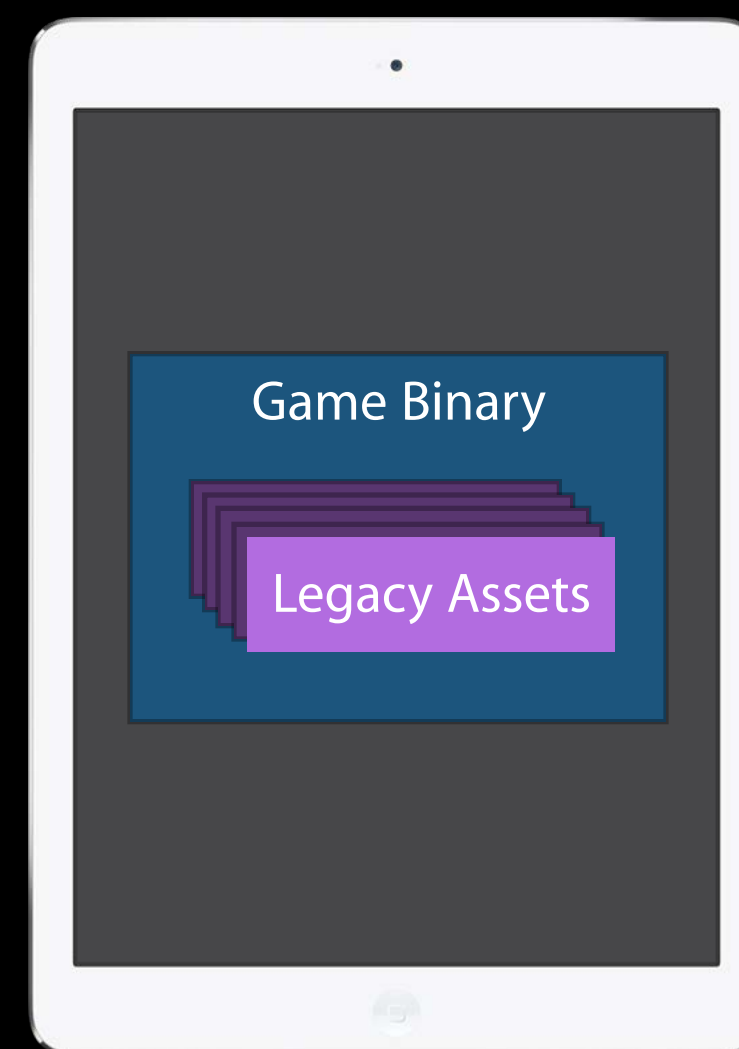




Legacy

Metal Capable

With App Thinning, only the assets applicable to the device are downloaded on install



# Capability Matrix

512MB

1GB

2GB

---

Metal GPUFamily2

---

Metal GPUFamily1

---

OpenGL ES Legacy

---

# Capability Matrix

Typical normal map example

---

Metal GPUFamily1

---

OpenGL ES Legacy

---



# Capability Matrix

Typical normal map example

---

Metal GPUFamily1

512x512  
EAC

---

OpenGL ES Legacy

---

# Capability Matrix

Typical normal map example

---

Metal GPUFamily1

512x512  
EAC

---

OpenGL ES Legacy

512x512  
RG8

---

# Capability Matrix

Typical normal map example

---

Metal GPUFamily1

512x512  
EAC

---

OpenGL ES Legacy

512x512  
RG8

---

# Capability Matrix

Extended normal map example

Format	512MB	1GB	2GB	
				256x256 RG8
				512x512 RG8
Metal GPUFamily2				512x512 EAC
				512x512 ASTC
Metal GPUFamily1				
				1024x1024 ASTC
OpenGL ES Legacy				



# Capability Matrix

Extended normal map example

Format	512MB	1GB	2GB	
Metal GPUFamily2			1024x1024 ASTC	256x256 RG8 512x512 RG8
Metal GPUFamily1				512x512 EAC 512x512 ASTC
OpenGL ES Legacy				

# Capability Matrix


Extended normal map example

Format	512MB	1GB	2GB	
Metal GPUFamily2		512x512 ASTC	1024x1024 ASTC	256x256 RG8 512x512 RG8 512x512 EAC
Metal GPUFamily1				
OpenGL ES Legacy				

# Capability Matrix

Extended normal map example

Format	512MB	1GB	2GB
Metal GPUFamily2		512x512 ASTC	1024x1024 ASTC
Metal GPUFamily1		512x512 EAC	
OpenGL ES Legacy			



# Capability Matrix

Extended normal map example

256x256  
RG8

Format	512MB	1GB	2GB
Metal GPUFamily2		512x512 ASTC	1024x1024 ASTC
Metal GPUFamily1		512x512 EAC	
OpenGL ES Legacy		512x512 RG8	



# Capability Matrix

Extended normal map example

Format	512MB	1GB	2GB
Metal GPUFamily2		512x512 ASTC	1024x1024 ASTC
Metal GPUFamily1		512x512 EAC	
OpenGL ES Legacy	256x256 RG8	512x512 RG8	









TestGPUData > TestGPUData > Assets.xcassets > NormalMaps

- TestGPUData
  - TestGPUData
    - AppDelegate.h
    - AppDelegate.m
    - ViewController.h
    - ViewController.m
    - Main.storyboard
    - Assets.xcassets**
    - LaunchScreen.storyboard
    - Info.plist
    - Supporting Files
    - TestGPUDataTests
    - TestGPUDataUITests
    - Products

- AppIcon
- NormalMaps**

### NormalMaps

 Any / Any	 Any / Metal 1v2	 Any / Metal 2v2
 1 GB / Any	 1 GB / Metal 1v2	 1 GB / Metal 2v2
Universal		

### Data Set

Name: NormalMaps

Device:  Universal  
 iPhone  
 iPad  
 Mac  
 Apple Watch

Memory:  1 GB  
 2 GB

Graphics:  Metal 1v2  
 Metal 2v2

### On Demand Resource Tags

Tags



No Matches







Show Slicing

- TestGPUData
  - TestGPUData
    - AppDelegate.h
    - AppDelegate.m
    - ViewController.h
    - ViewController.m
    - Main.storyboard
    - Assets.xcassets**
    - LaunchScreen.storyboard
    - Info.plist
    - Supporting Files
    - TestGPUDataTests
    - TestGPUDataUITests
    - Products

AppIcon

**NormalMaps**

### NormalMaps

 Any / Any	 Any / Metal 1v2	 Any / Metal 2v2
 1 GB / Any	 1 GB / Metal 1v2	 1 GB / Metal 2v2
Universal		

#### Data Set

Name: NormalMaps

Device:  Universal  
 iPhone  
 iPad  
 Mac  
 Apple Watch

Memory:  1 GB  
 2 GB

Graphics:  Metal 1v2  
 Metal 2v2

#### On Demand Resource Tags

Tags







No Matches



- TestGPUData
  - TestGPUData
    - AppDelegate.h
    - AppDelegate.m
    - ViewController.h
    - ViewController.m
    - Main.storyboard
    - Assets.xcassets**
    - LaunchScreen.storyboard
    - Info.plist
    - Supporting Files
    - TestGPUDataTests
    - TestGPUDataUITests
    - Products

- AppIcon
- NormalMaps**

### NormalMaps

 Any / Any	 Any / Metal 1v2	 Any / Metal 2v2
 1 GB / Any	 1 GB / Metal 1v2	 1 GB / Metal 2v2
Universal		

### Data Set

Name: NormalMaps

Device:  Universal  
 iPhone  
 iPad  
 Mac  
 Apple Watch

Memory:  1 GB  
 2 GB

Graphics:  Metal 1v2  
 Metal 2v2

On Demand Resource Tags

Tags



No Matches









TestGPUData > TestGPUData > Assets.xcassets > NormalMaps

- TestGPUData
  - TestGPUData
    - AppDelegate.h
    - AppDelegate.m
    - ViewController.h
    - ViewController.m
    - Main.storyboard
    - Assets.xcassets**
    - LaunchScreen.storyboard
    - Info.plist
    - Supporting Files
    - TestGPUDataTests
    - TestGPUDataUITests
    - Products

AppIcon

**NormalMaps**

### NormalMaps

 Any / Any	 Any / Metal 1v2	 Any / Metal 2v2
 512x512 RG8 1 GB / Any	 512x512 EAC 1 GB / Metal 1v2	 1 GB / Metal 2v2
Universal		

### Data Set

Name: NormalMaps

Device:  Universal  
 iPhone  
 iPad  
 Mac  
 Apple Watch

Memory:  1 GB  
 2 GB

Graphics:  Metal 1v2  
 Metal 2v2

### On Demand Resource Tags

Tags

No Matches

Show Slicing

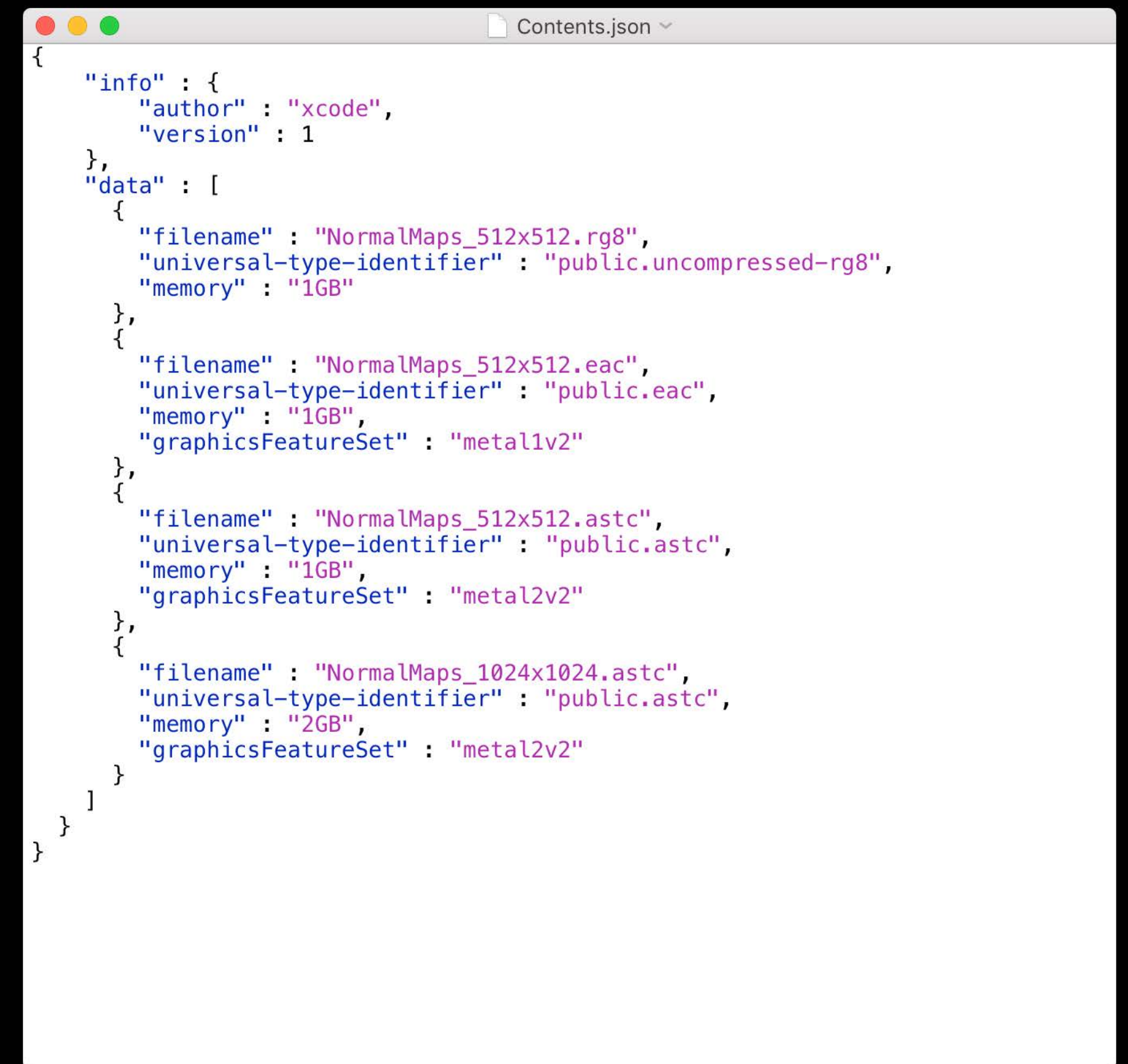


# Custom Tools Pipelines

# Custom Tools Pipelines

Publicly documented JSON file format

Easily integrated into custom toolchain



```
{
  "info" : {
    "author" : "xcode",
    "version" : 1
  },
  "data" : [
    {
      "filename" : "NormalMaps_512x512.rg8",
      "universal-type-identifier" : "public.uncompressed-rg8",
      "memory" : "1GB"
    },
    {
      "filename" : "NormalMaps_512x512.eac",
      "universal-type-identifier" : "public.eac",
      "memory" : "1GB",
      "graphicsFeatureSet" : "metal1v2"
    },
    {
      "filename" : "NormalMaps_512x512.astc",
      "universal-type-identifier" : "public.astc",
      "memory" : "1GB",
      "graphicsFeatureSet" : "metal2v2"
    },
    {
      "filename" : "NormalMaps_1024x1024.astc",
      "universal-type-identifier" : "public.astc",
      "memory" : "2GB",
      "graphicsFeatureSet" : "metal2v2"
    }
  ]
}
```

# Retrieving Named Data

# Retrieving Named Data

NSDataAsset class provides correctly matched data resource from Asset Catalog

# Retrieving Named Data

NSDataAsset class provides correctly matched data resource from Asset Catalog

```
#import <UIKit/NSDataAsset.h>
```



# Retrieving Named Data

NSDataAsset class provides correctly matched data resource from Asset Catalog

```
#import <UIKit/NSDataAsset.h>
```

```
NSDataAsset *asset = [[NSDataAsset alloc] initWithName:@"NormalMaps"];
```

```
NSData *data = asset.data;
```

## Art Pipeline

512x512  
RG8

512x512  
ASTC

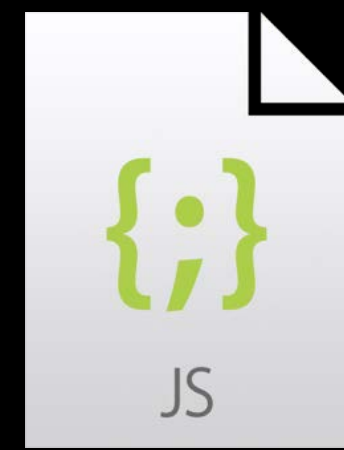
512x512  
EAC

## Art Pipeline

512x512  
RG8

512x512  
ASTC

512x512  
EAC

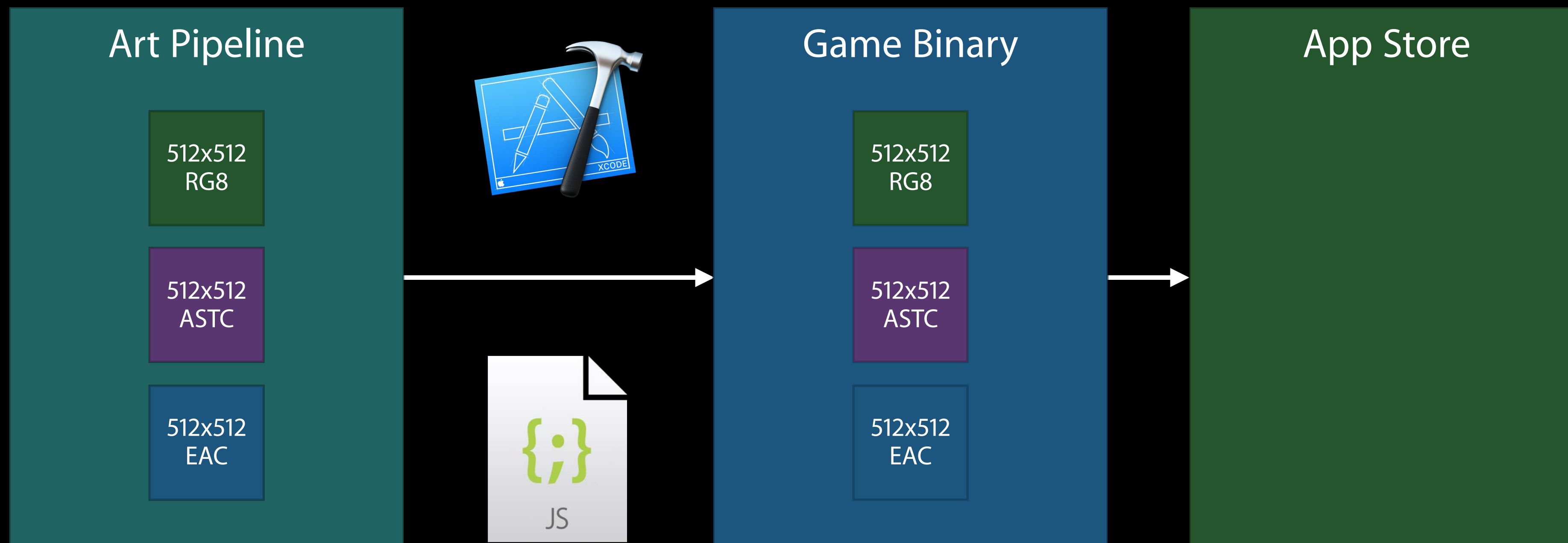


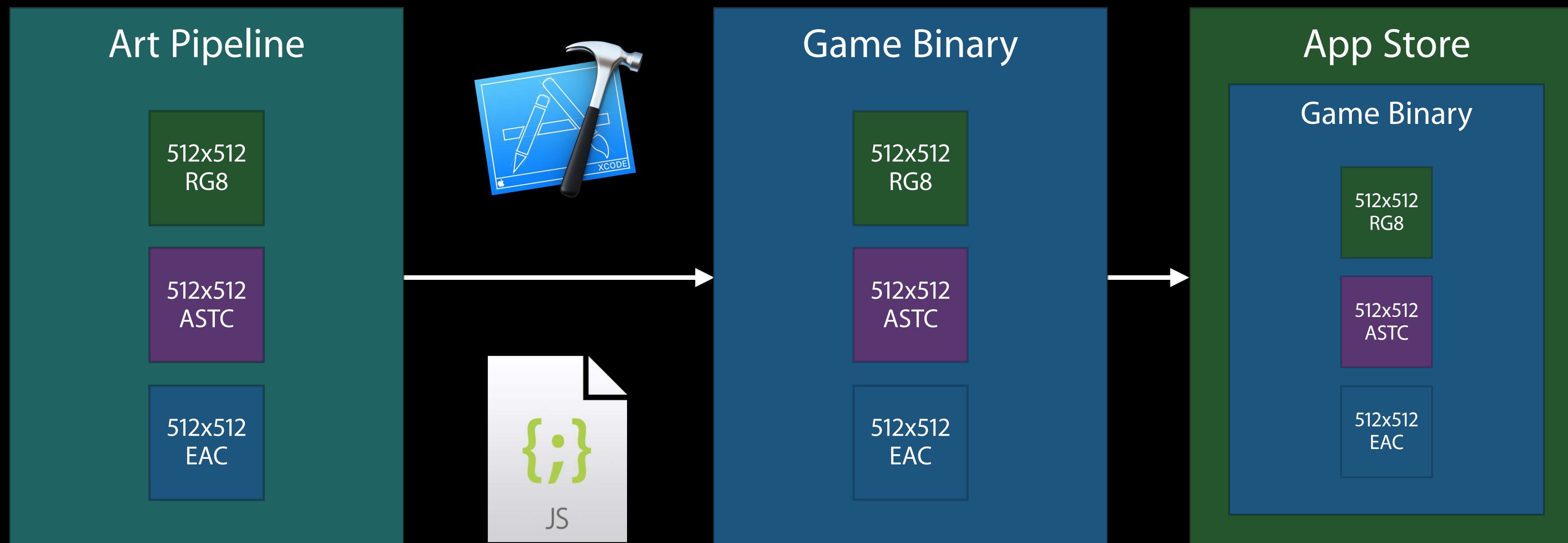
## Game Binary

512x512  
RG8

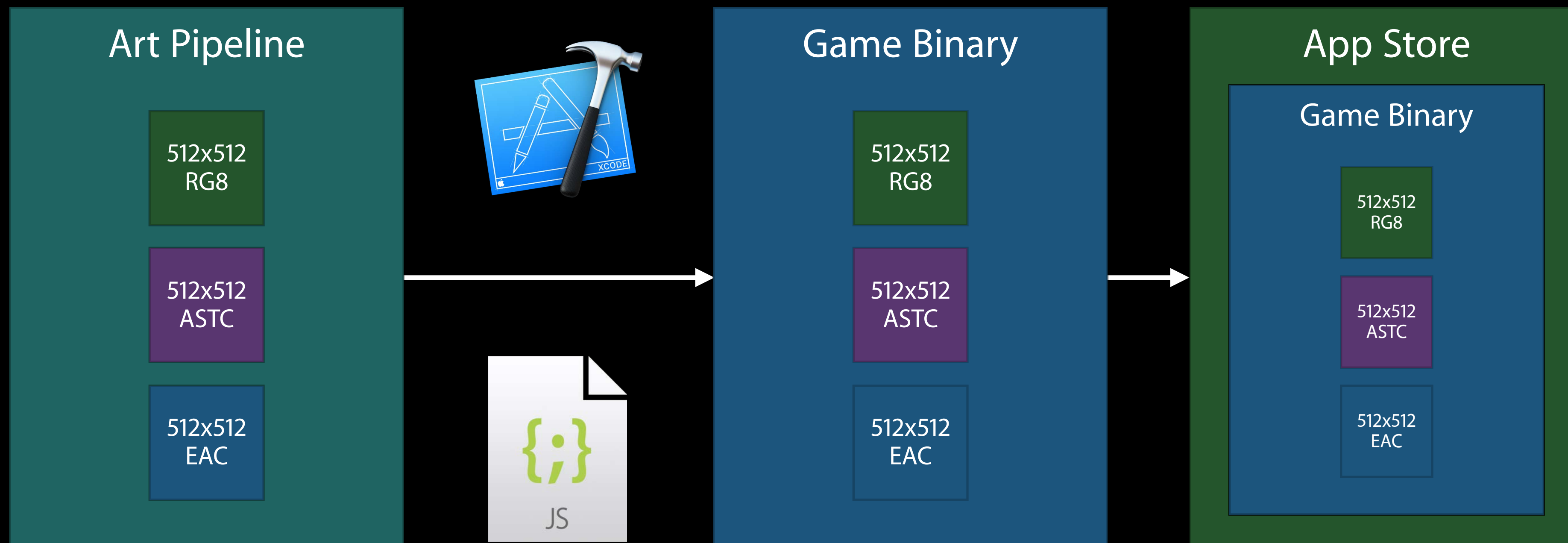
512x512  
ASTC

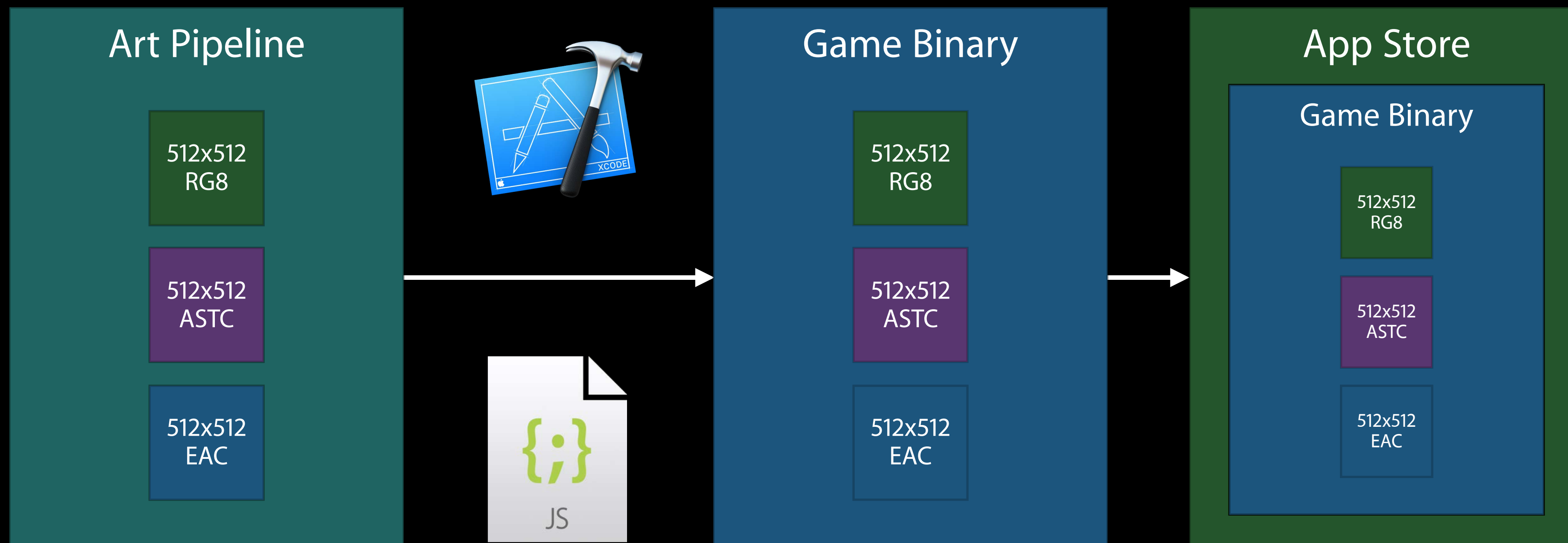
512x512  
EAC



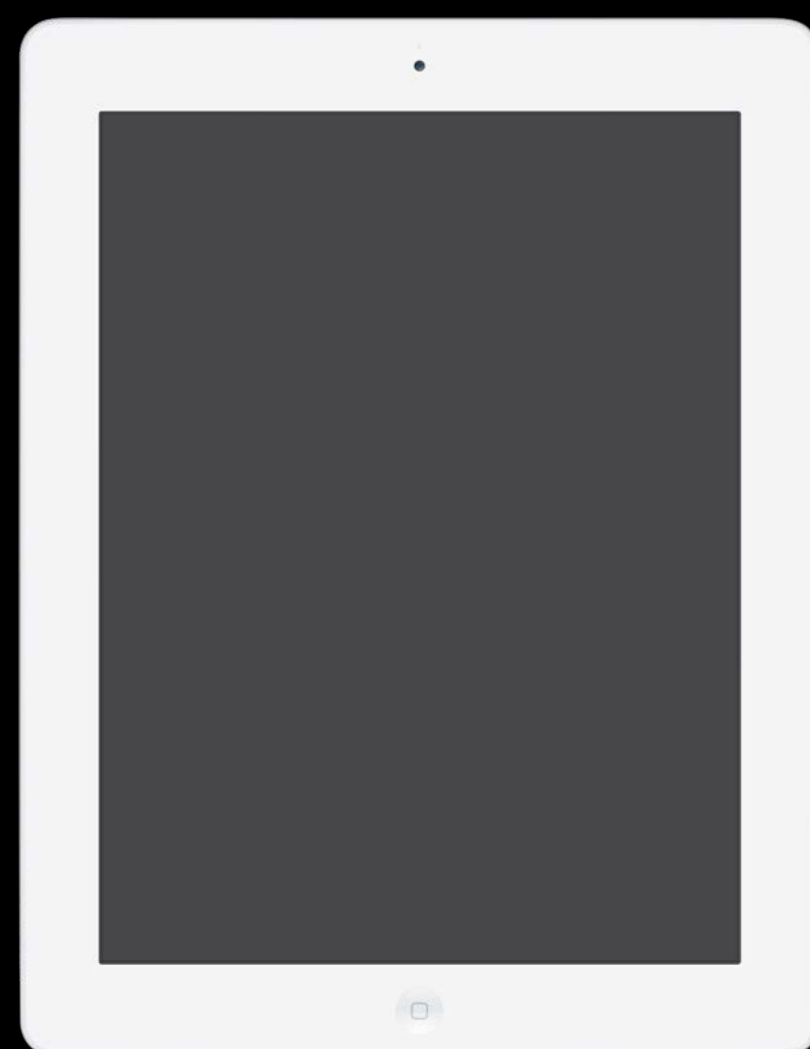








iPad 2

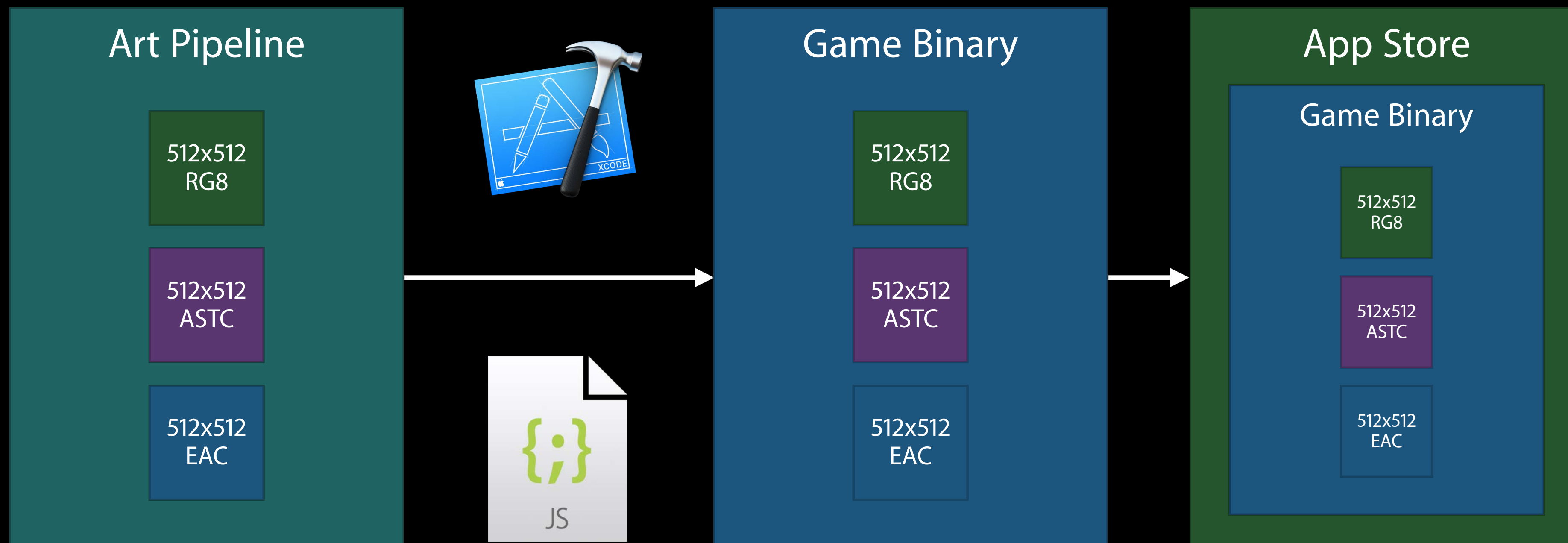


iPad Air

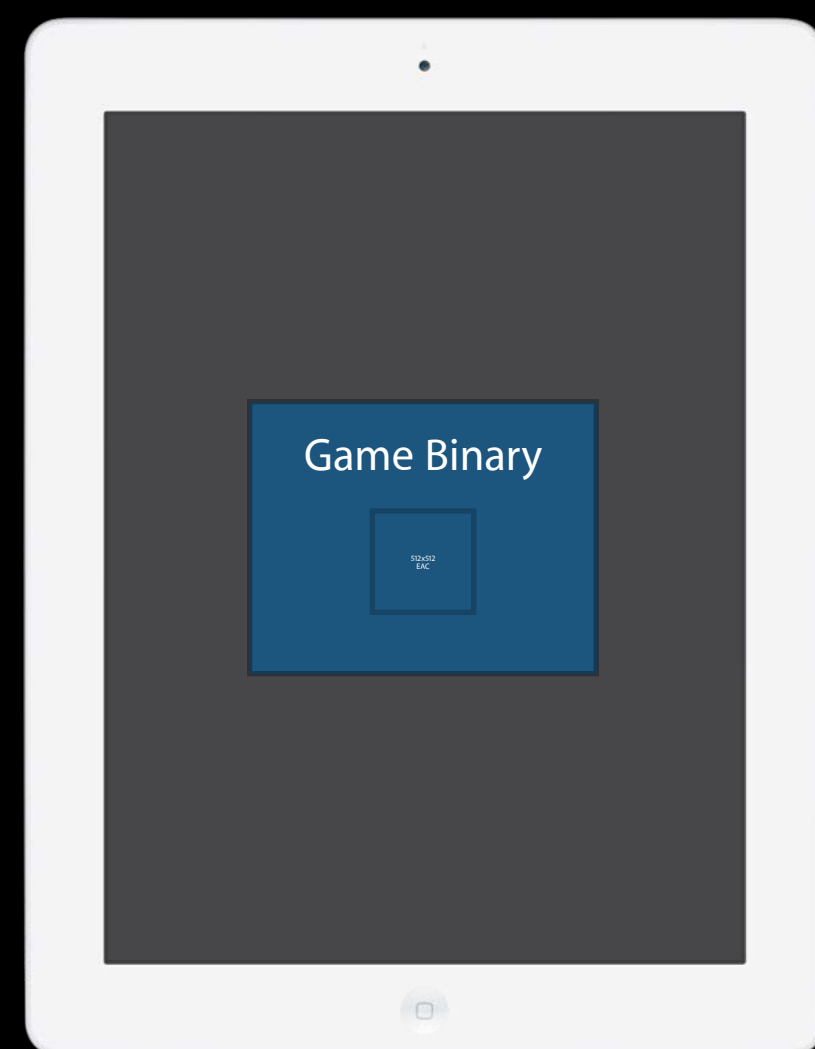


iPad Air 2

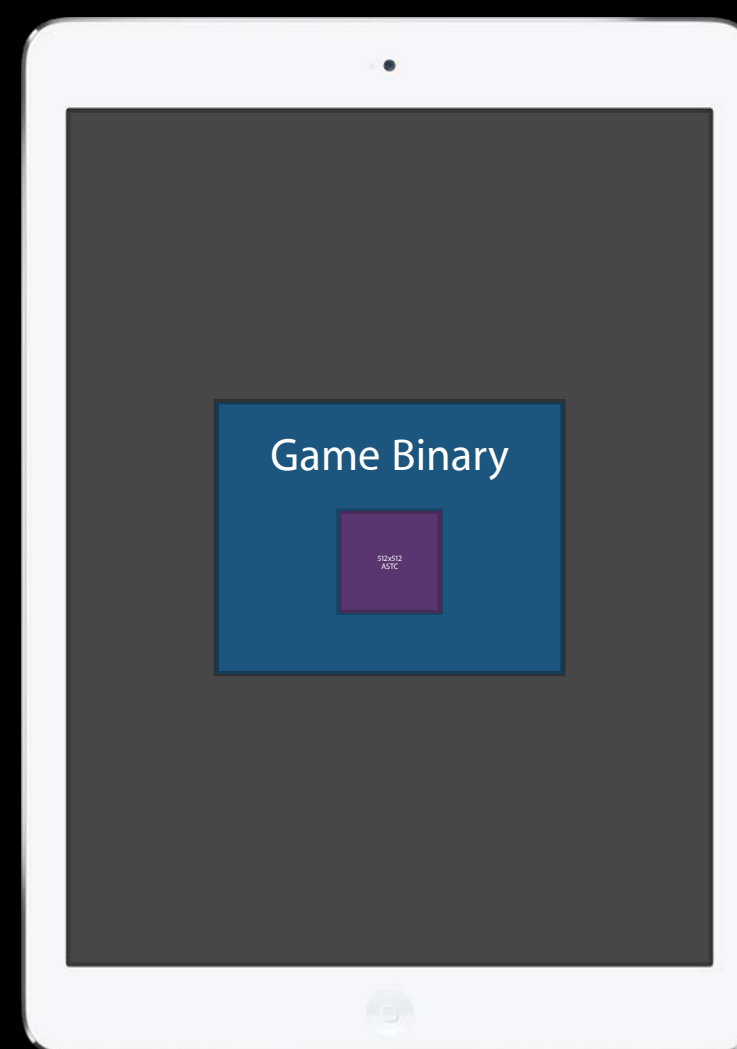




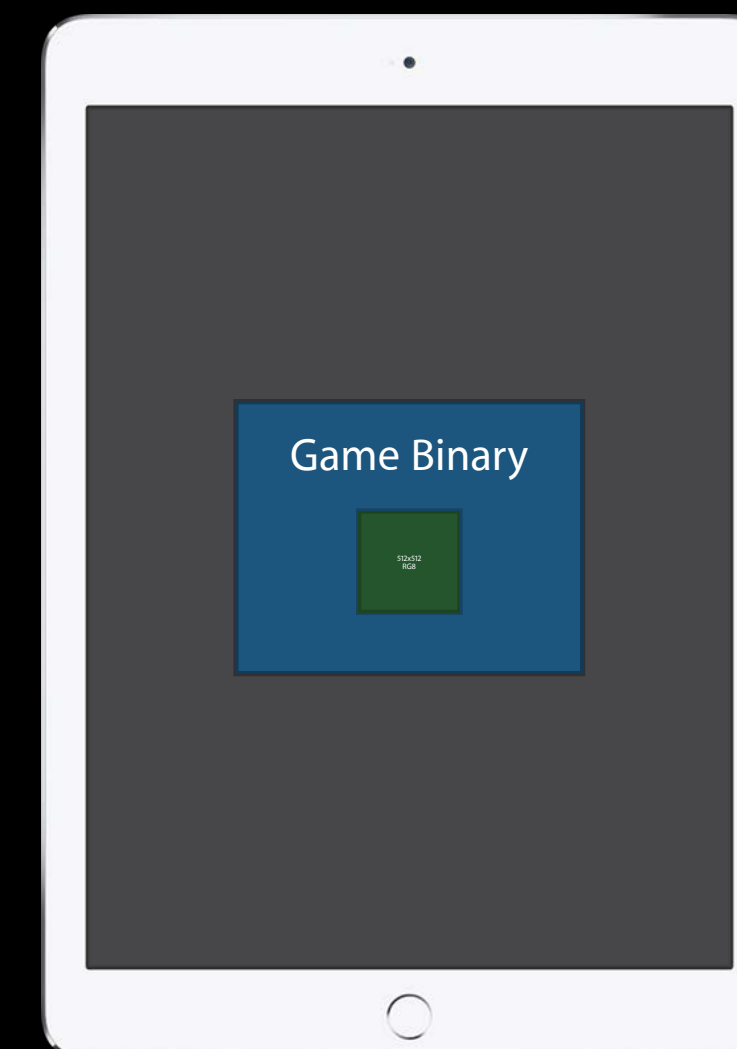
iPad 2



iPad Air



iPad Air 2



# Summary

Developers are using Metal to create next-generation games and professional apps

Metal now available for OS X

New Xcode Metal tools

New API features in iOS 9 and OS X

Support Metal-specific assets with App Thinning

# More Information

Metal Documentation and Videos

<http://developer.apple.com/metal>

Apple Developer Forums

<http://developer.apple.com/forums>

Developer Technical Support

<http://developer.apple.com/support/technical>

General Inquiries

Allan Schaffer, Game Technologies Evangelist

[aschaffer@apple.com](mailto:aschaffer@apple.com)



# Related Sessions

---

What's New in Metal, Part 2

Mission

Thursday 9:00AM

---

Metal Performance Optimization Techniques

Pacific Heights

Friday 11:00AM

---

# Labs

---

Metal Lab

Graphics, Games,  
and Media Lab C

Wednesday 11:00AM

---

Metal Lab

Graphics, Games,  
and Media Lab D

Friday 12:00PM

---

 WWDC 15