Graphics and Games

#WWDC15

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
- Metal
- Shaders

Metal Performance Optimization Techniques

- Metal System Trace T
- Metal Best P

Metal at WWDC

What's New in Metal, Part 1

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

What's New in Metal, Part 2

- Introducing MetalKit
- Metal Perfomance Shaders

Metal Performance Optimization Techniques

- Metal System Trace T
- Metal Best P

Metal at WWDC

What's New in Metal, Part 1

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

What's New in Metal, Part 2

- Introducing
- Metal
- 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

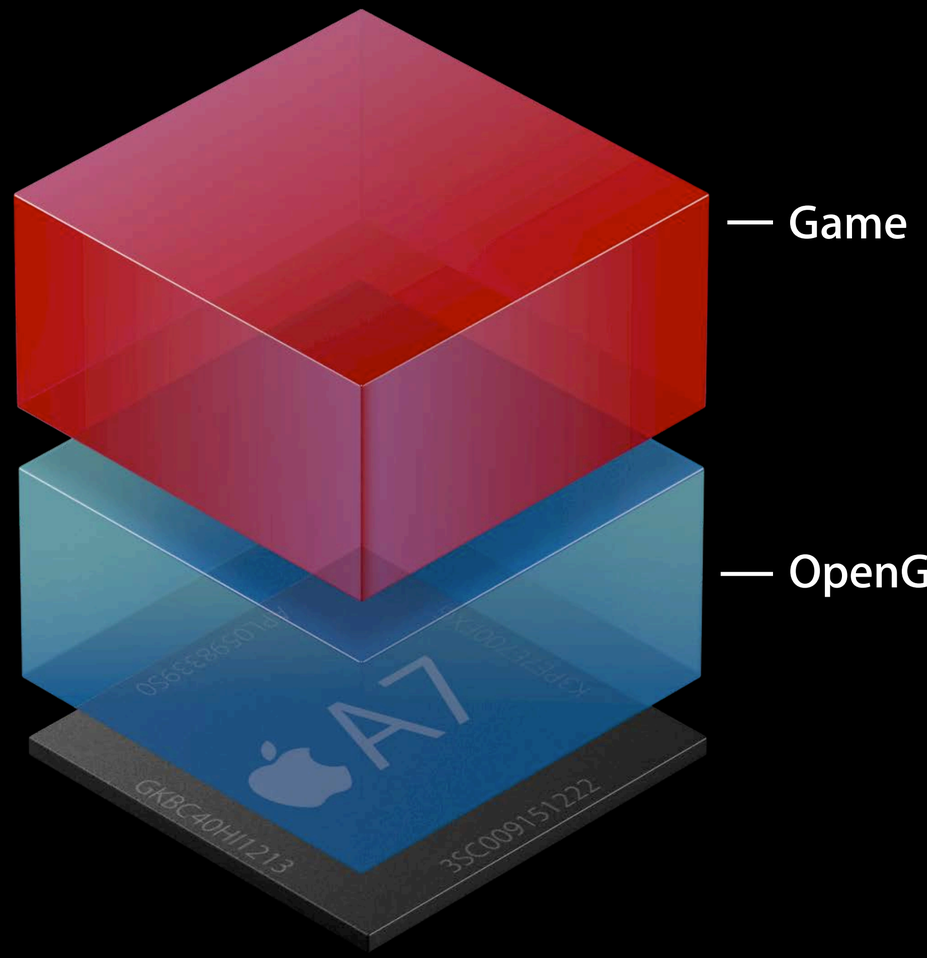
Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading



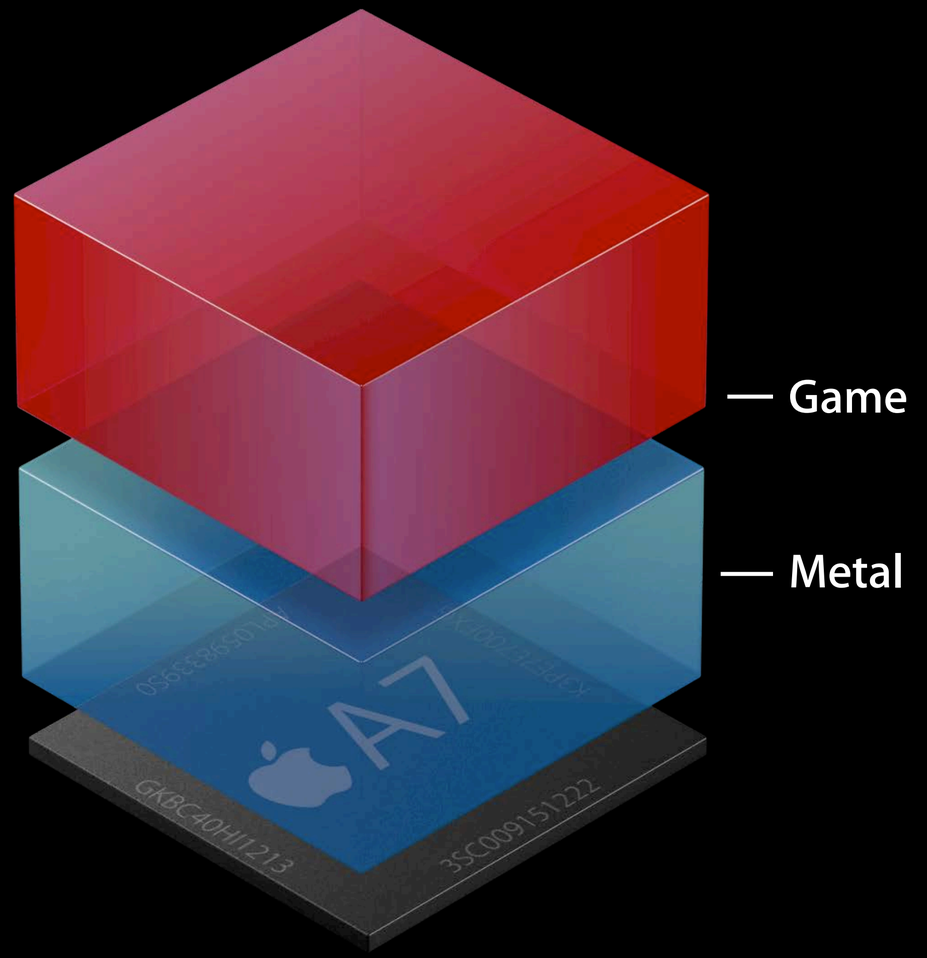
Metal

Dramatically reduced overhead

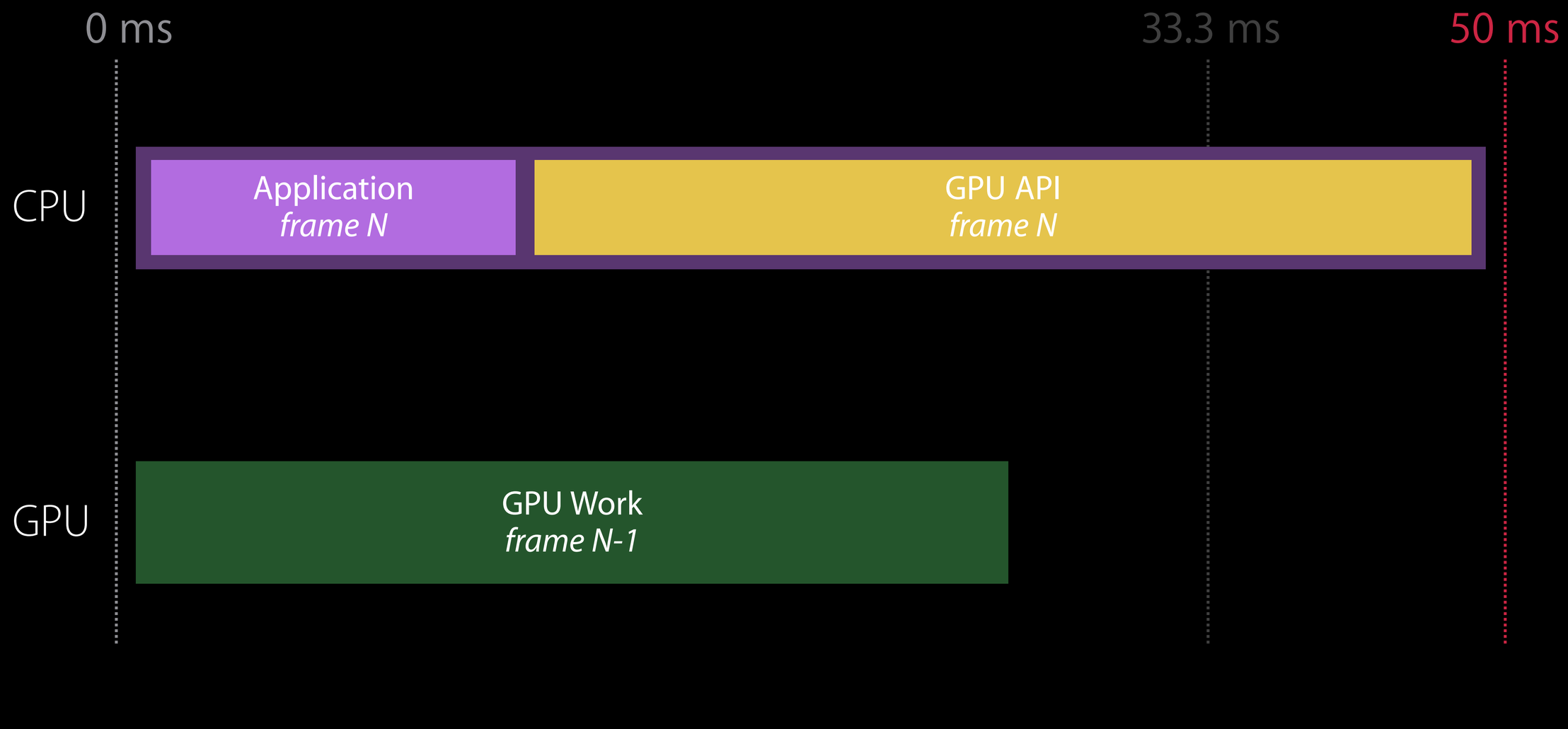
Precompiled shaders

Graphics and compute

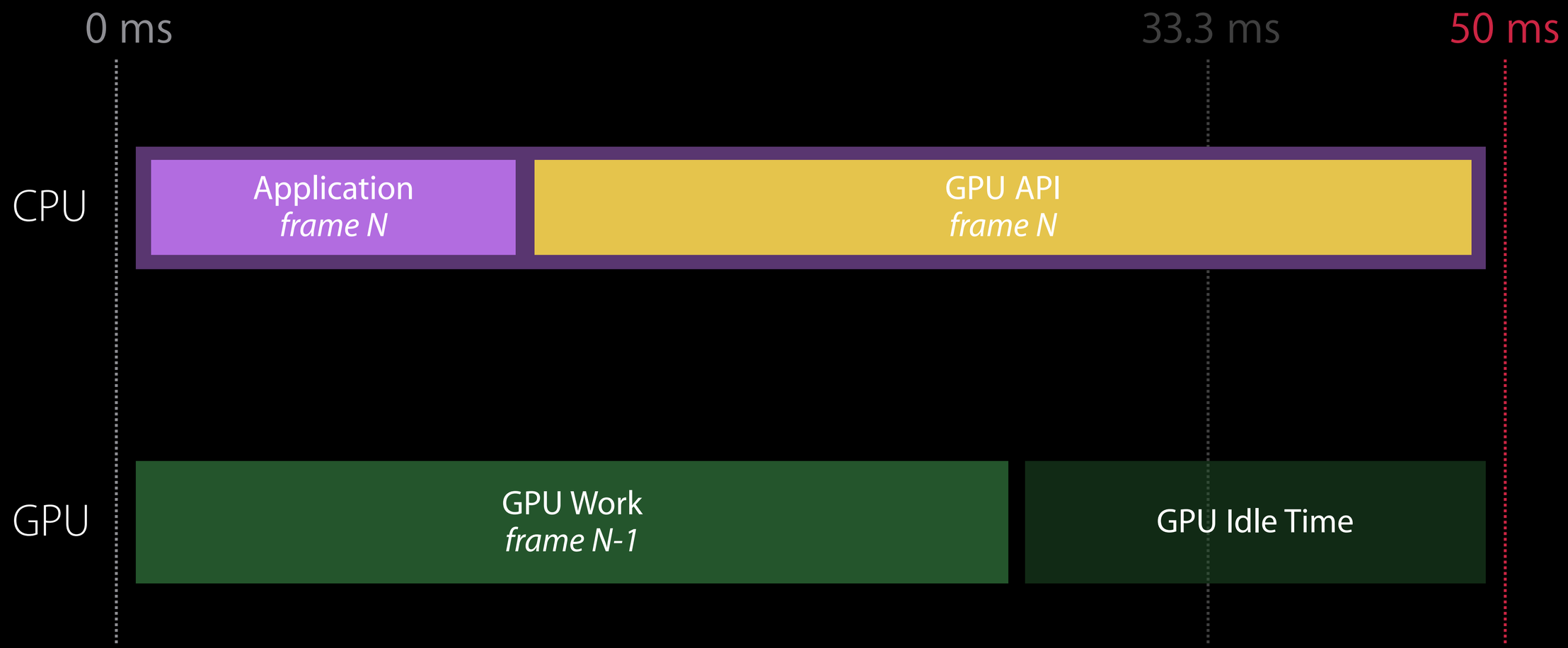
Efficient multithreading



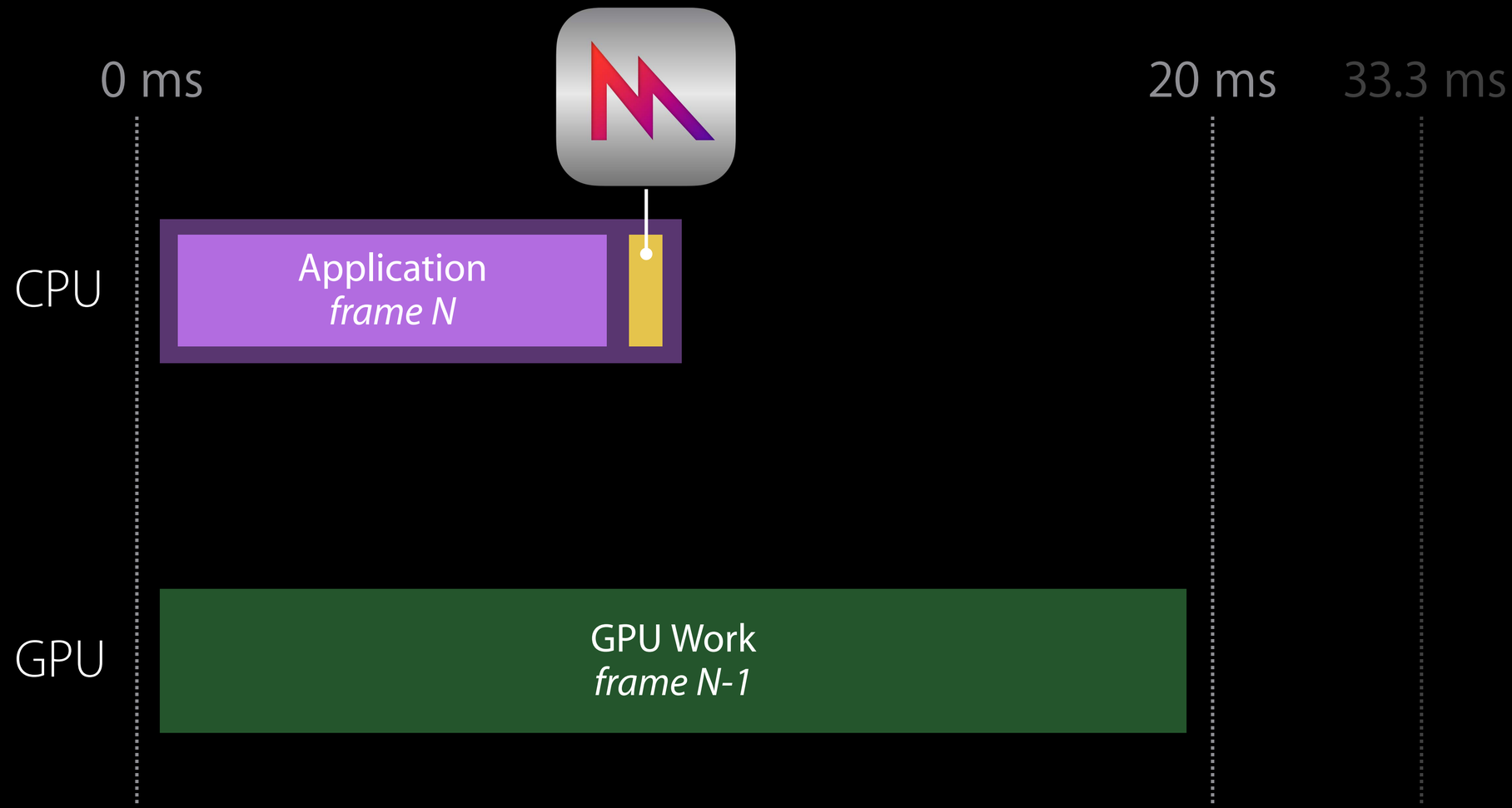
Frame Time with CPU as the Bottleneck



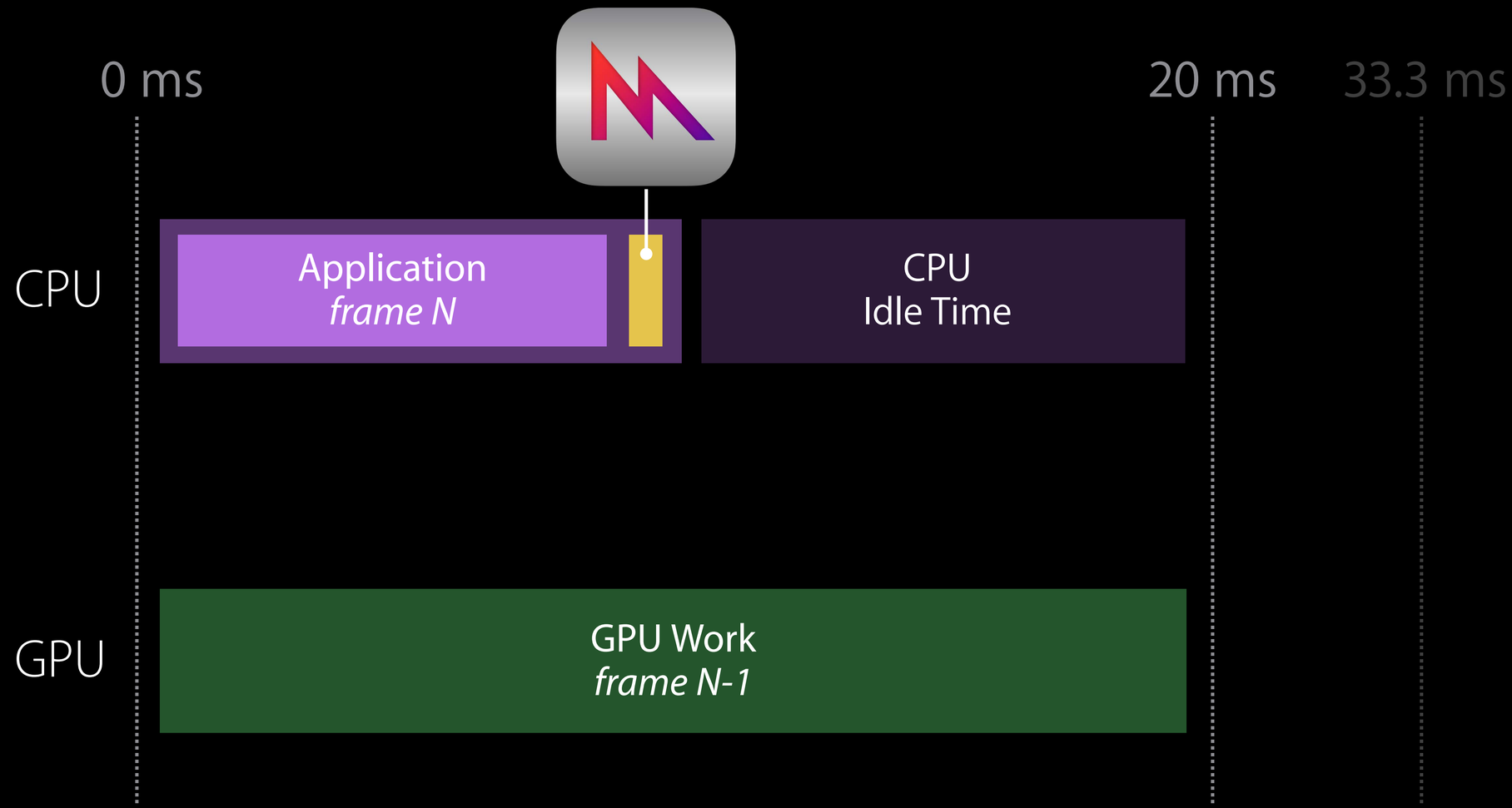
Frame Time with CPU as the Bottleneck



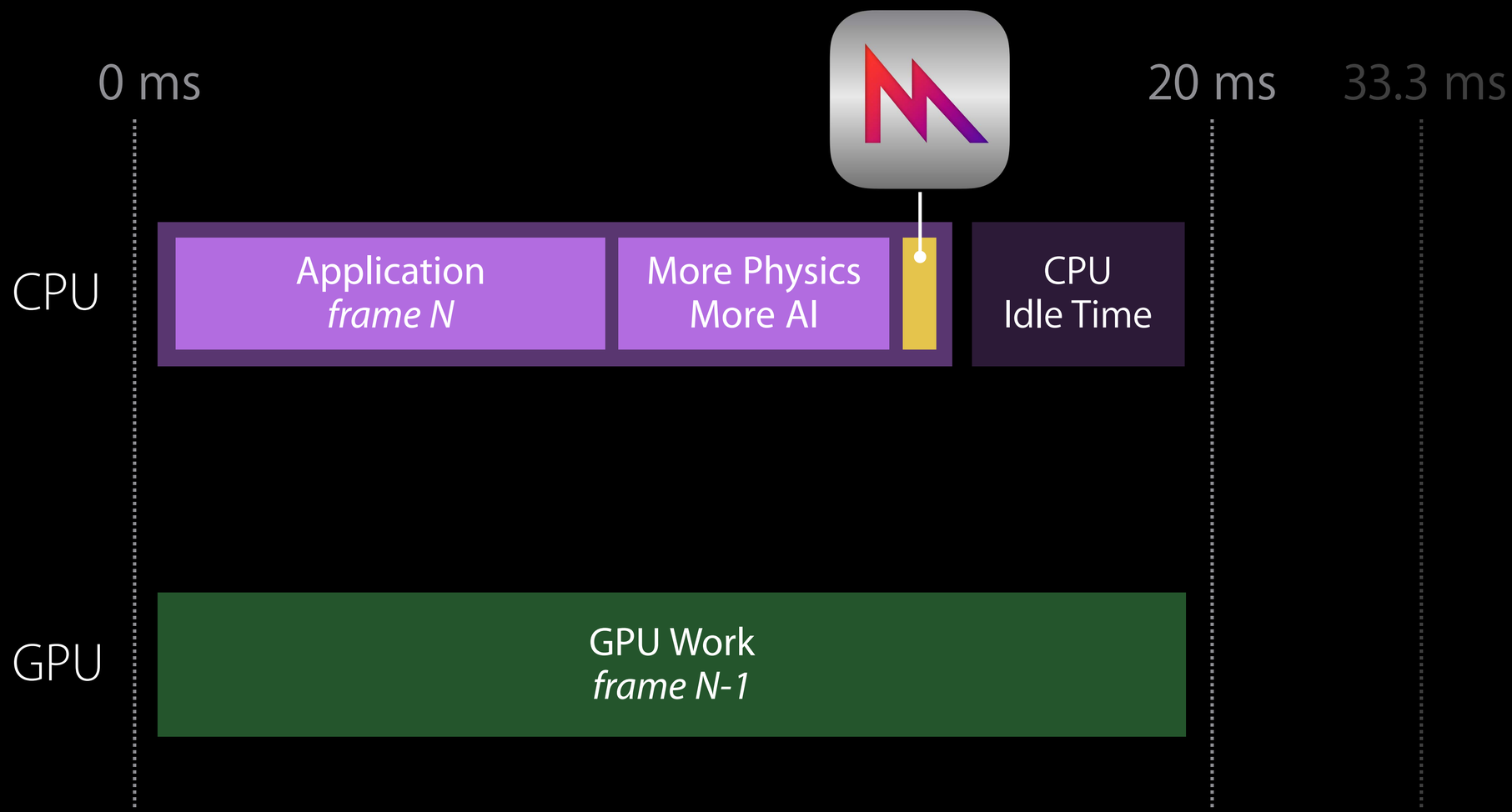
Metal Reduces GPU API Overhead



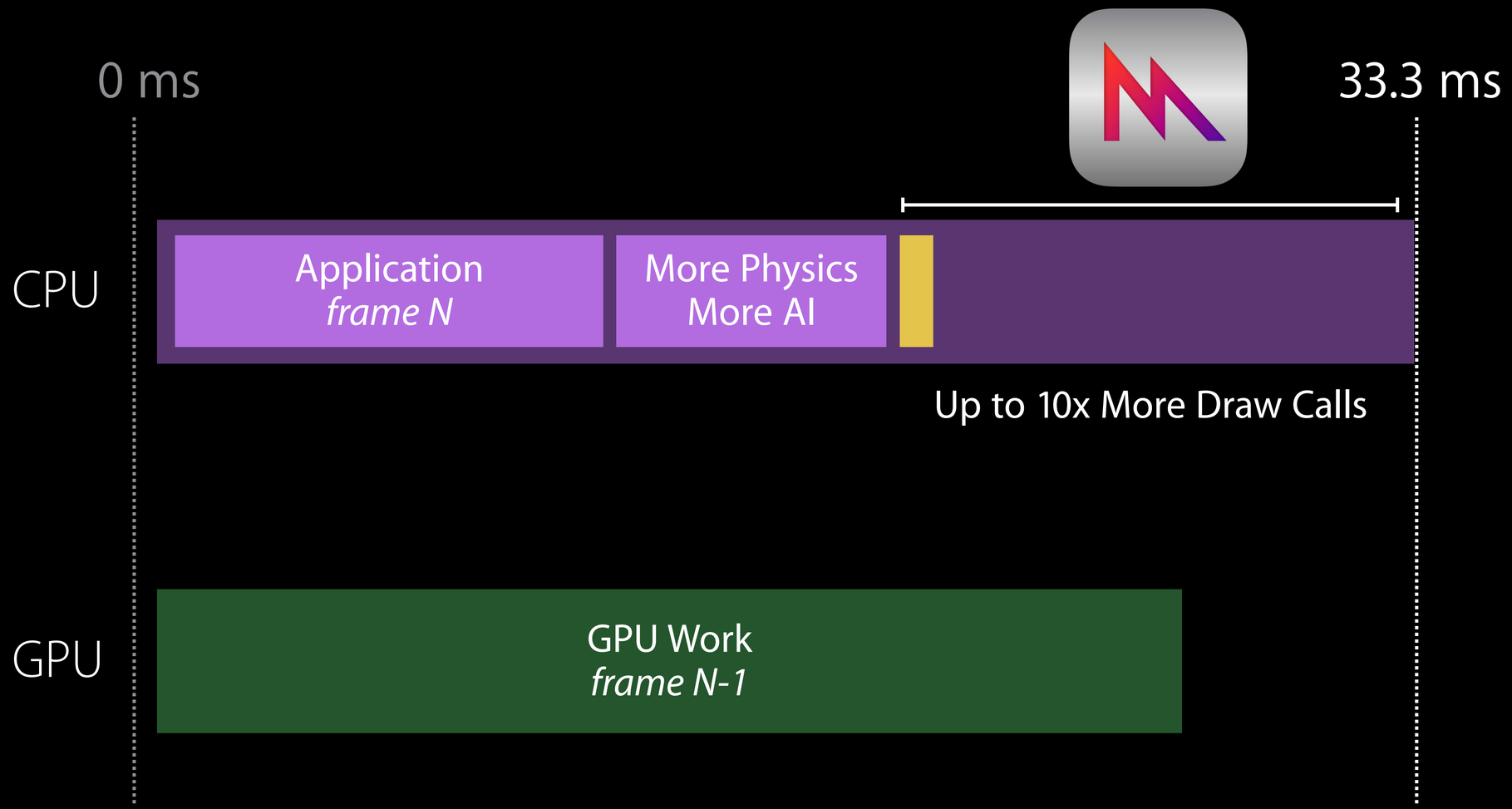
Metal Reduces GPU API Overhead



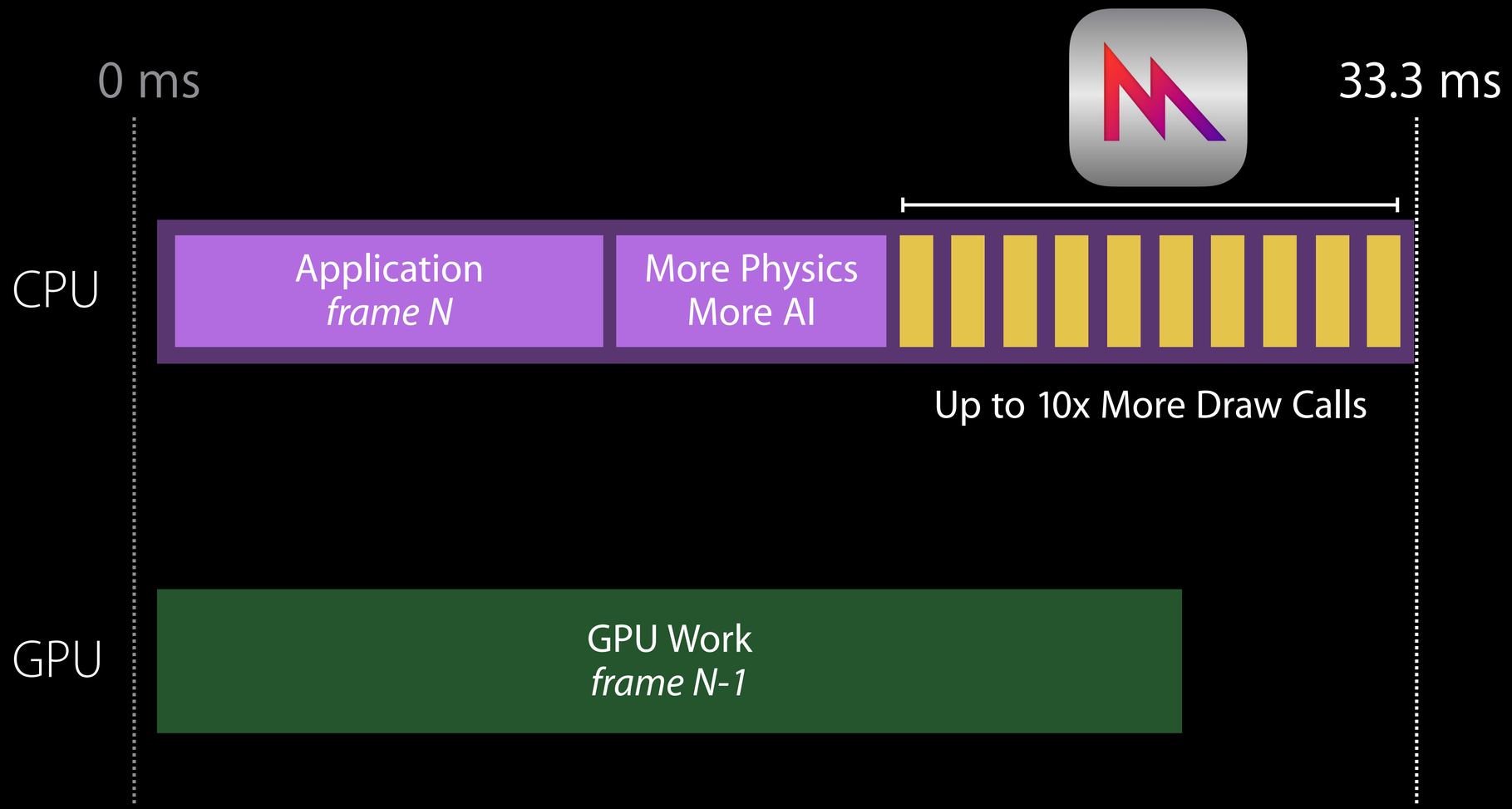
Improve Your Game



Or Issue More Draw Calls



Or Issue More Draw Calls



Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading

Build Time	Load Time	Draw Time
"Never"	Infrequent	1000s per Frame
		Shader Compilation
		State Validation

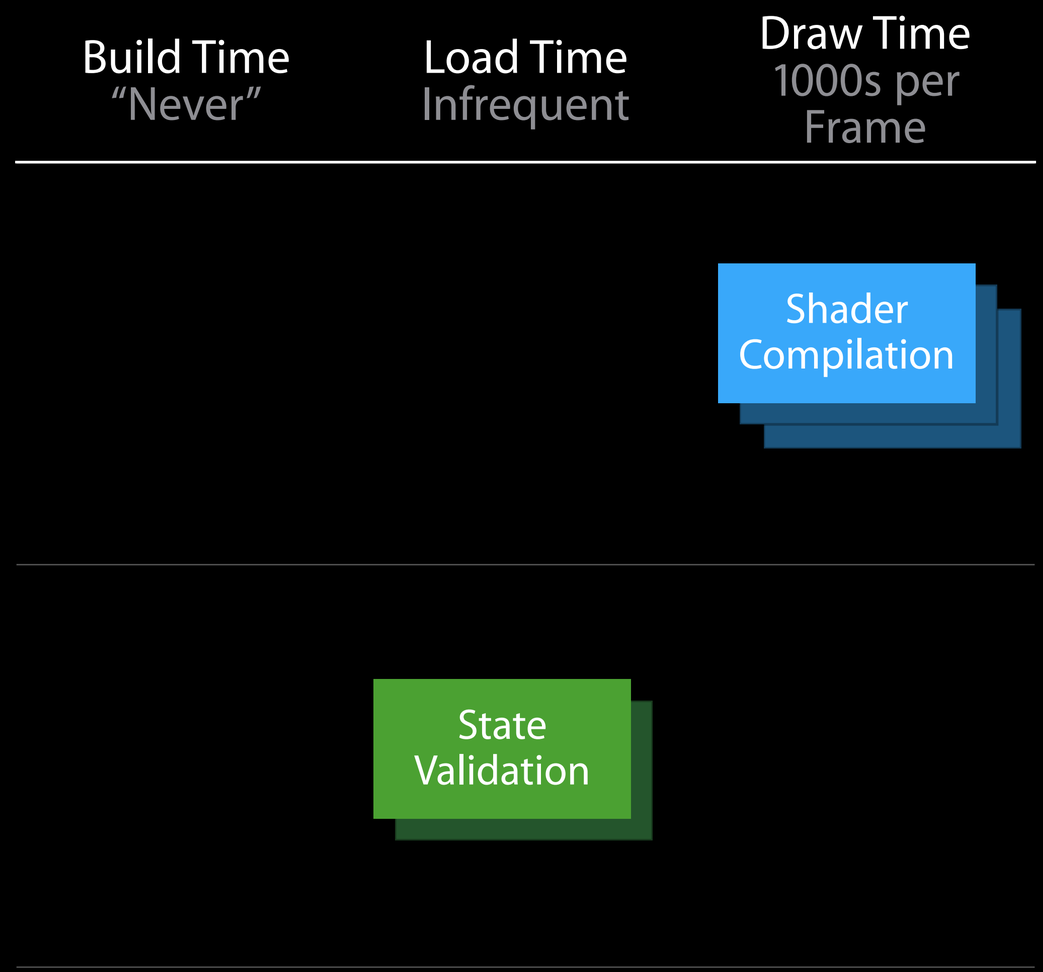
Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading



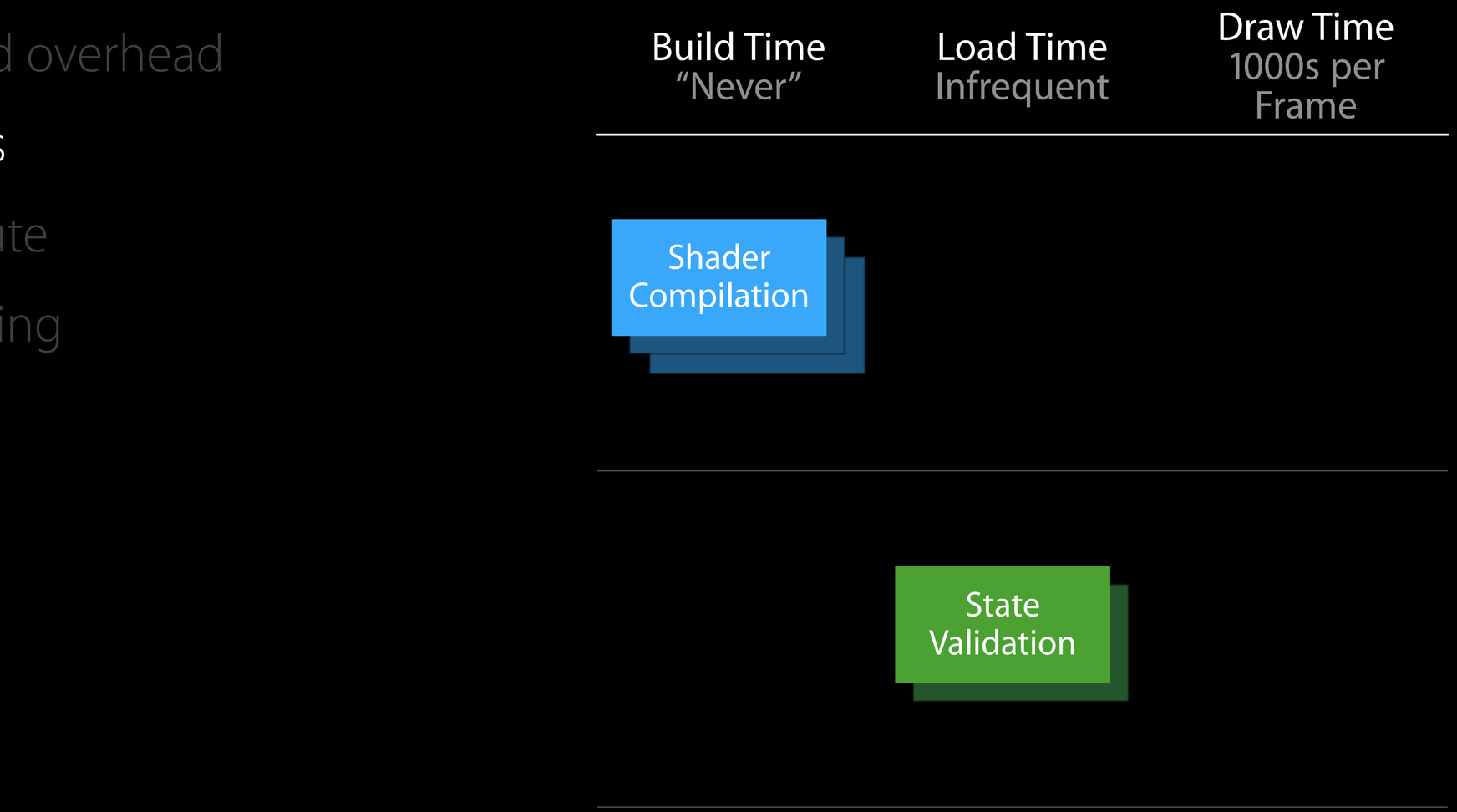
Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading



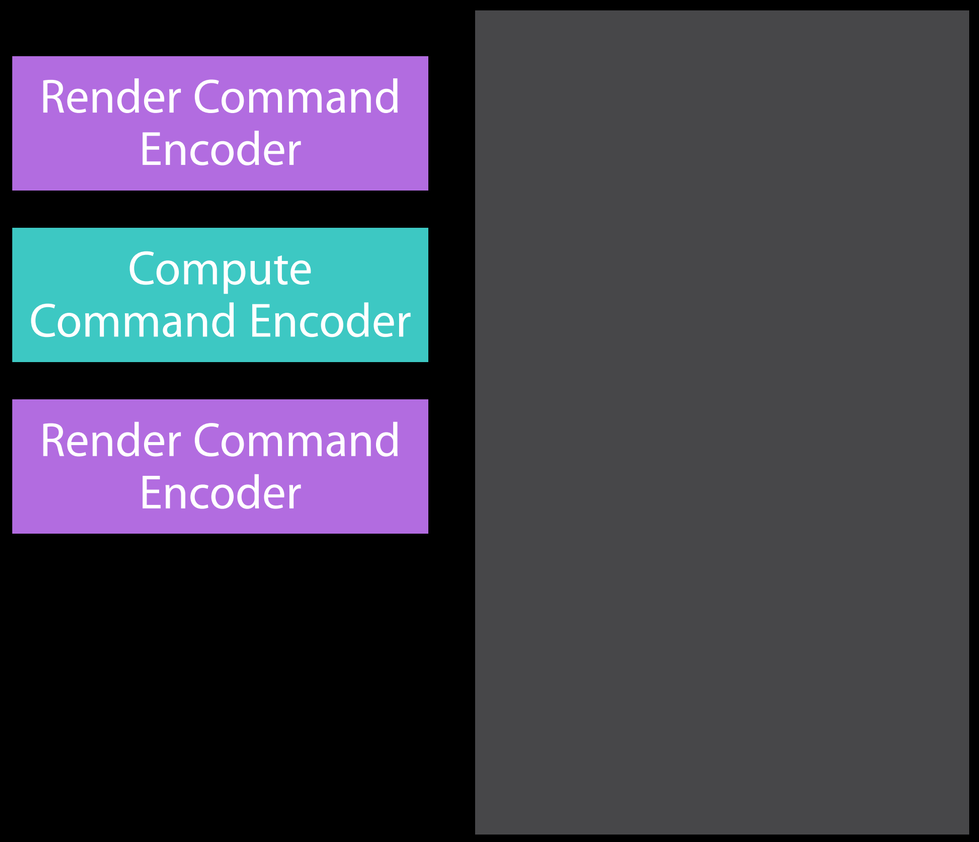
Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading



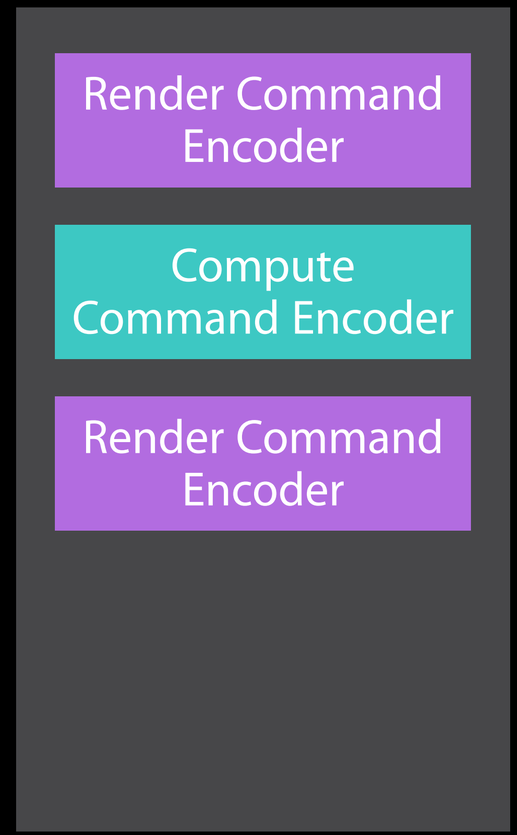
Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading



Command Buffer

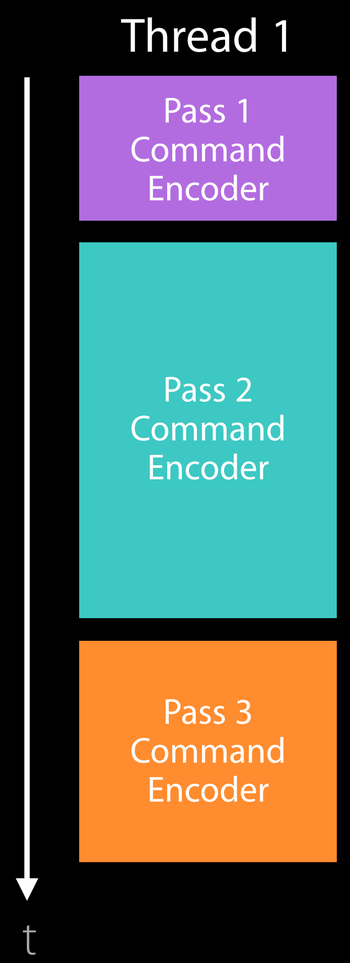
Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading



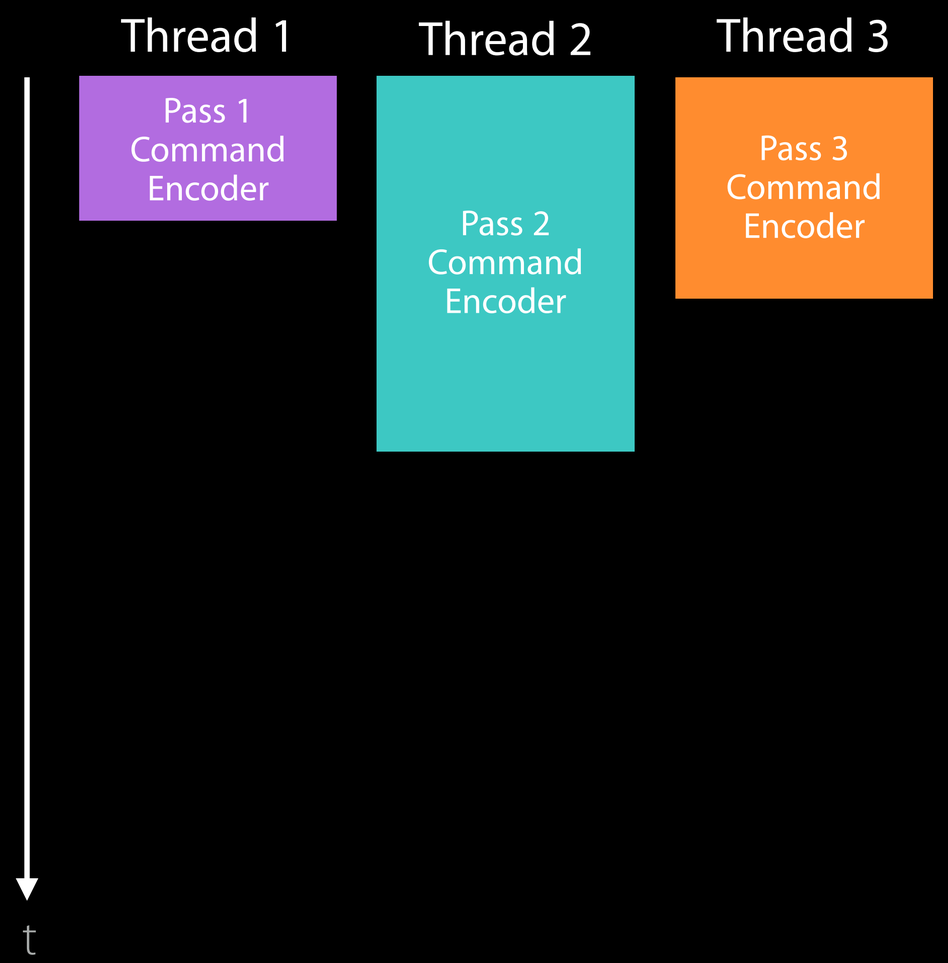
Metal

Dramatically reduced overhead

Precompiled shaders

Graphics and compute

Efficient multithreading




MARKET
MEAT


VAIN
GLORY
ベイン
グローリー


ASPHALT 8
AIRBORNE


Cancel
Undo
Warp
Apply

EA

GAMELOFT

Disney

Illusion Labs

PIXELMATOR TEAM

naturalmotion GAMES

EPIC GAMES

UBISOFT

DIGITAL LEGENDS ENTERTAINMENT

SUPER EVIL MEGACORP

REBELLION

ROVIO STARS

unity

ATYPICAL GAMES

HEAVY IRON STUDIOS

Midoki

CD PROJEKT RED

FUSION INTERACTIVE

505 GAMES

PROLETARIAT

FUERO GAMES

VECTOR UNIT

zynga

bE HAVIOUR

SQUARE ENIX

NIVAL

TECHLAND

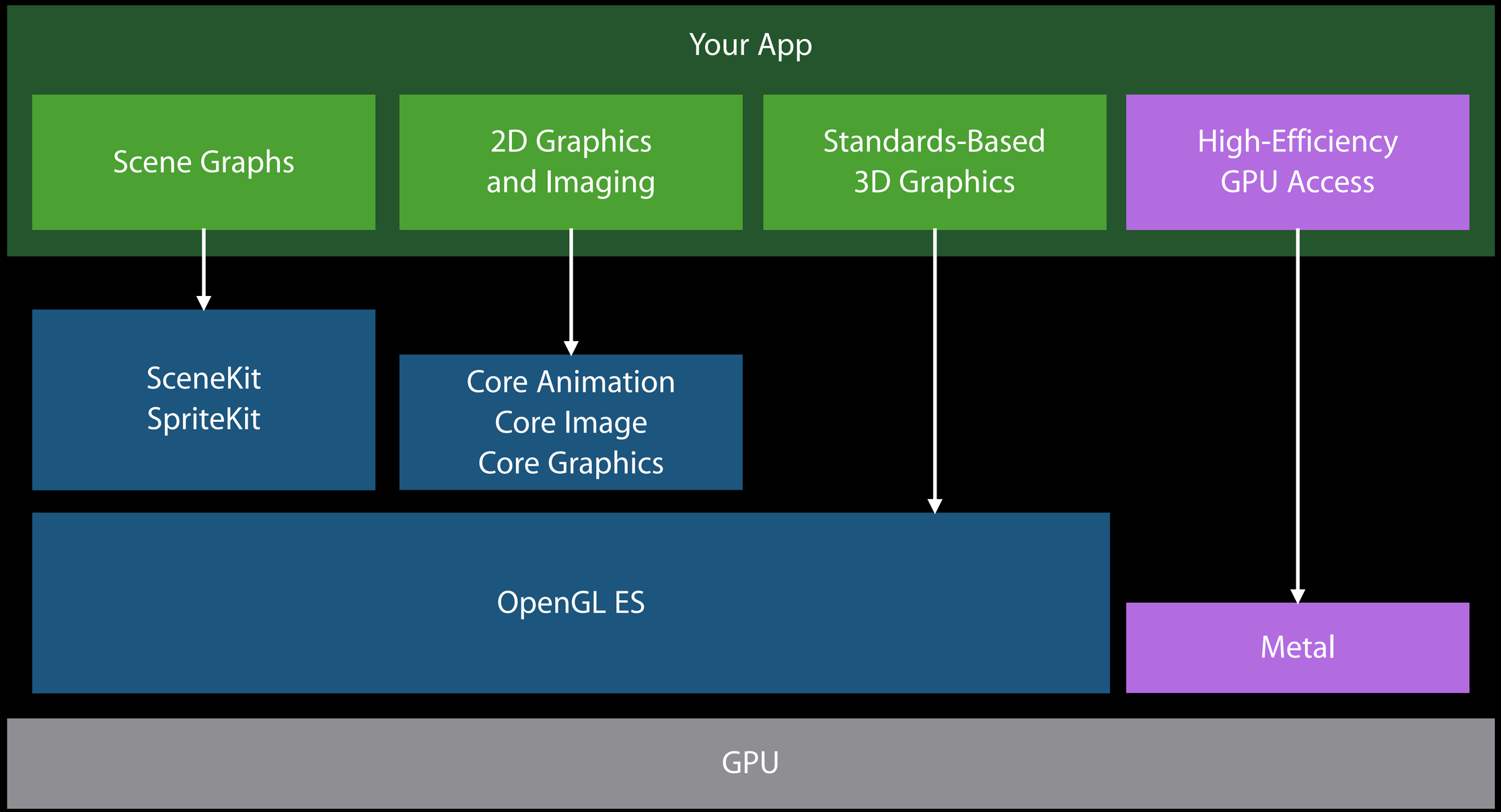
IWA GAMES

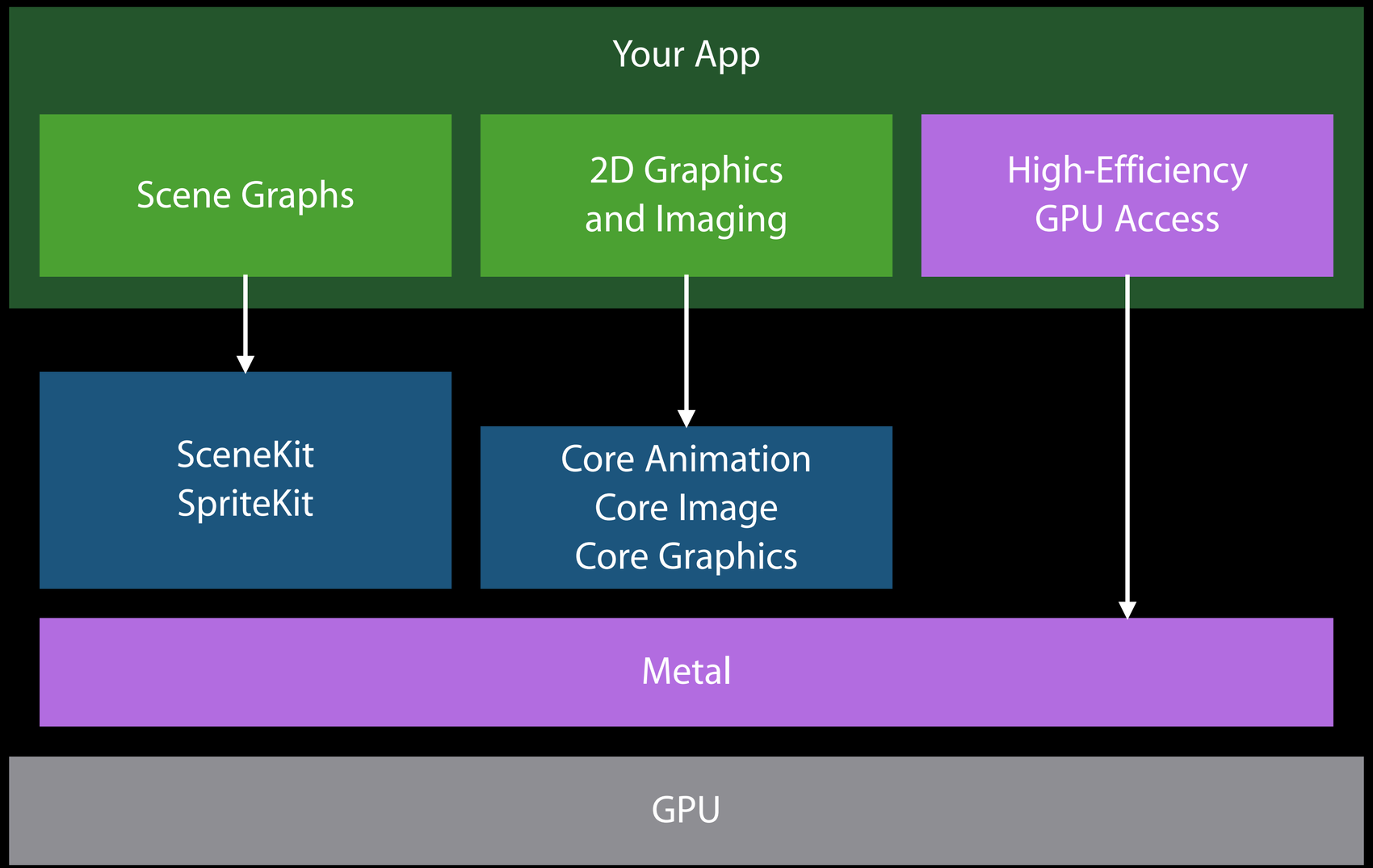
pixelbite


unity


UNREAL
ENGINE


FROSTBITE™


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


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


iPad
9:41 AM
100%
FaceTime
Calendar
Thursday
16
Photos
Camera
Contacts
Clock
Maps
Videos
Notes
Reminders
Photo Booth
Newsstand
iTunes Store
App Store
Settings
Extras
Messages
Mail
Safari
9:41 AM
100%
Messages
Calendar
Photos
Camera
Weather
Clock
Maps
Videos
Notes
Reminders
Stocks
Game Center
Newsstand
iTunes Store
App Store
iBooks
Health
Passbook
Settings
Phone
Mail
Safari
Music


iPad
9:41 AM
100%
FaceTime
Calendar
Thursday
16
Photos
Contacts
Clock
Maps
Notes
Reminders
Photo Booth
Newsstand
iTunes Store
App Store
Settings
Extras
Messages
Mail
Safari
Weather
Stocks
Health
Passbook
Phone
Tuesday
Videos
Game Center
iBooks
Camera
Music

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

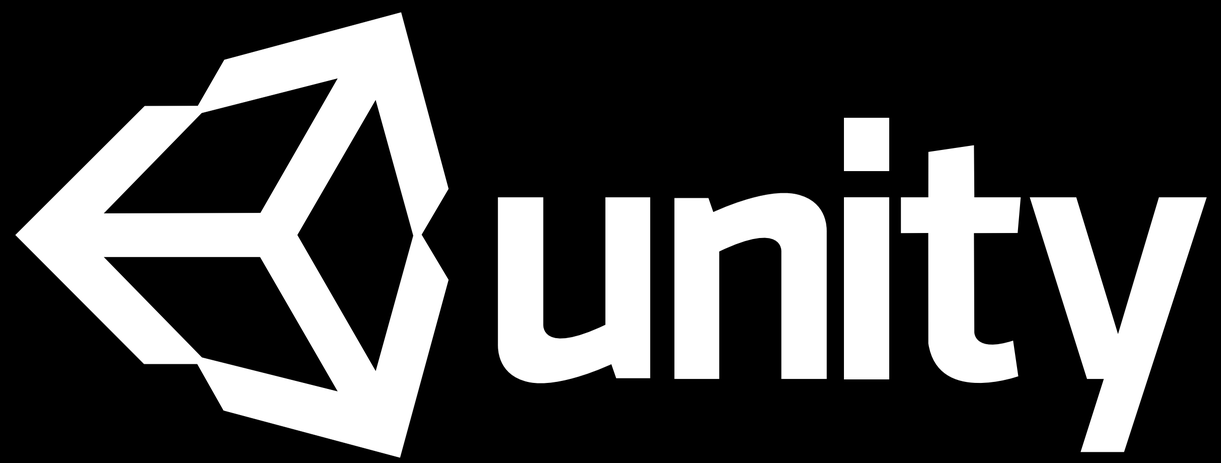
- Device selection
- Support for Discrete Memory

Metal OS X

Minimal code change required for existing iOS applications

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


EPIC
GAMES


unity

The_Viking_Village.unity - trunk - PC, Mac & Linux Standalone

Center Local

Layers Default Account

Hierarchy

Create All

Content
Reflection Probes
WorldBoundaries
Water
Light Probes
Occlusion Area
Wind Zone
Directional light
QualityManager
Canvas
EventSystem
WaterReflectionSceneCamera
WaterReflectio
WaterReflectio
WaterReflectio
AccessibleVolu
MouseLock
WaterReflectio

Scene Game

Shaded 2D Gizmos All

Persp

Viking Village

unity

Project

Create

Favorites
All Mate
All Mode
All Prefa
All Scrip

Assets
Animatio
Editor
Lighting
Materials
Models
Prefabs
Scenes
Scripts

Shaders Standard Ass... Textures

ProjectSettings.asset

Inspector

PlayerSettings

Company Name Unity Technologies
Product Name Viking Village

Default Icon None (Texture2

Default Cursor None (Texture2

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

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 GBObjects: 28088

2K

unity

BLIZZARD ENTERTAINMENT

FERAL
MAKE YOUR PLAY

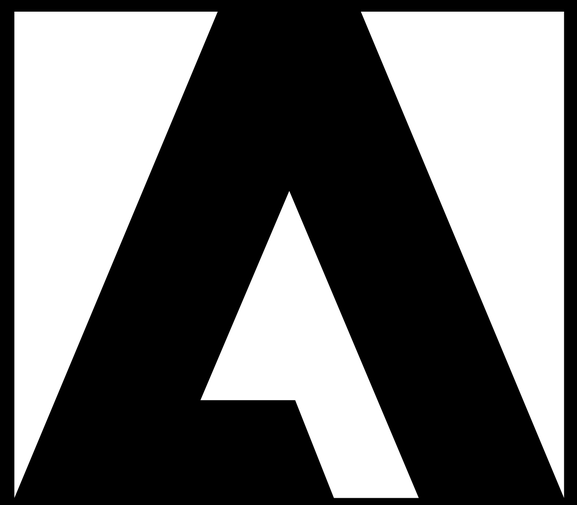
UNREAL ENGINE

THE FOUNDRY.

CAMPO SANTO

AUTODESK.

aspyr



Adobe

THE FOUNDRY.

Adopting Metal on OS X

The Foundry

Jack Greasley

Courtesy of Framestore, Heyday films, and StudioCanal

© The Witcher 3. Image courtesy of Platige Image.

Courtesy of Adidas

MODO®

Model F2 Sculpt | Items | Action Center | Symmetry | Falloff | Snapping | Select Through | Work Plane | Render F9 | Render Window | Sketchfab

3D View | UV Texture View | Render Preset Browser | Gradient Editor | Schematic +

Camera (2) | Advanced | Ray GL: Off

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

Channels 6

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

Audio | Graph Editor | Animated | 171 | Play | Settings

Play Forwards

Item List | Images | Vertex M... +

Filter Items | Add Item | F

Name

- SLS6_v0011.lxo*
  - Even Lighting Preset Se
    - 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 Luminousity (Textur
  - 007 Red (Texture)
  - 5X Luminousity (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

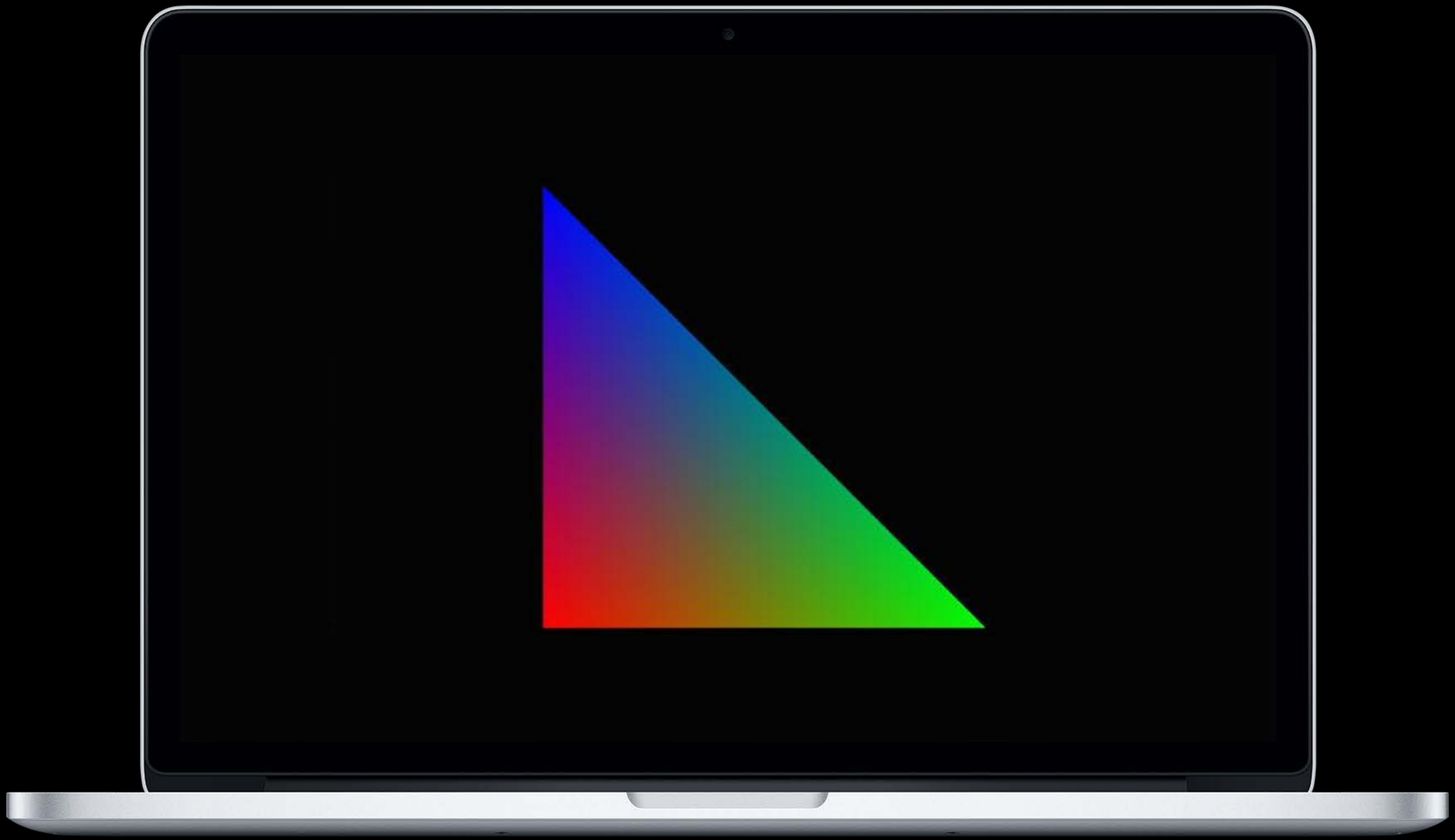
Film Back

Film Back Preset	(custom)
Film Width	36 mm
Height	24 mm
Film Offset X	0 m
Y	0 m
Film Fit	Fill

Camera View | Camera Effects | Display | Assembly | User Channels | Tags

Command

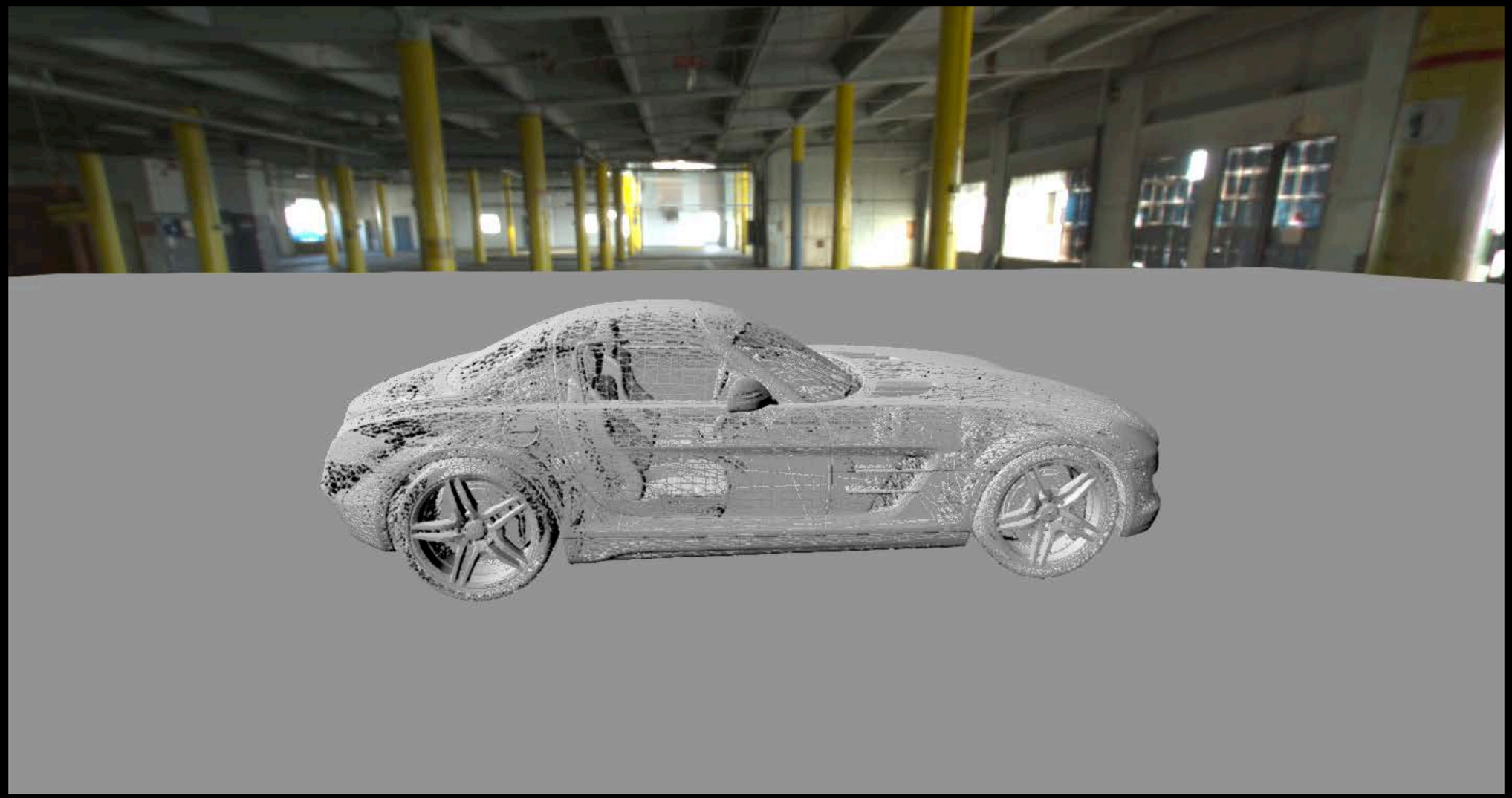

Finder File Edit View Go Window Help
Thu 9:41 AM
OCT
16



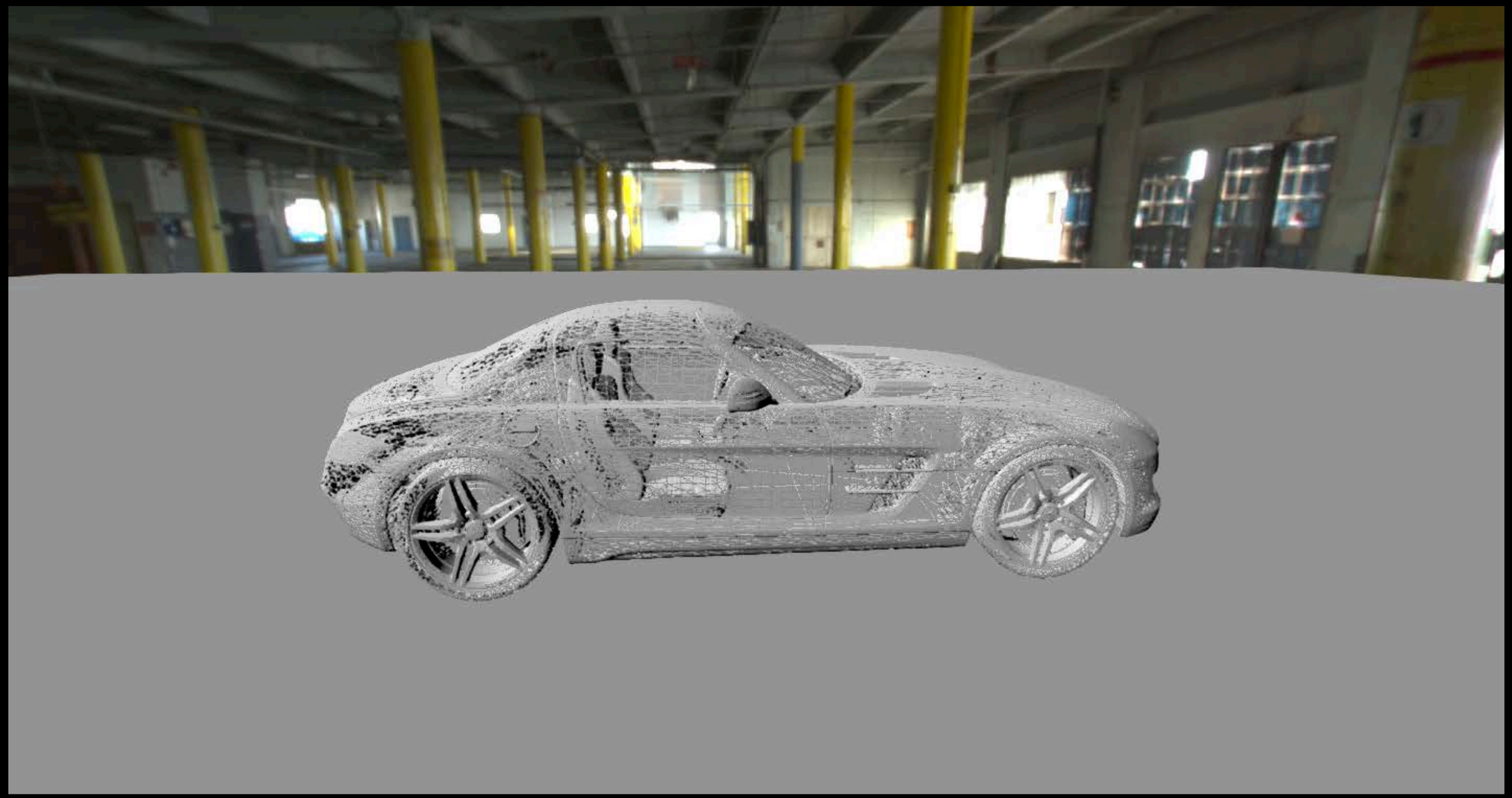
Day One

Day One

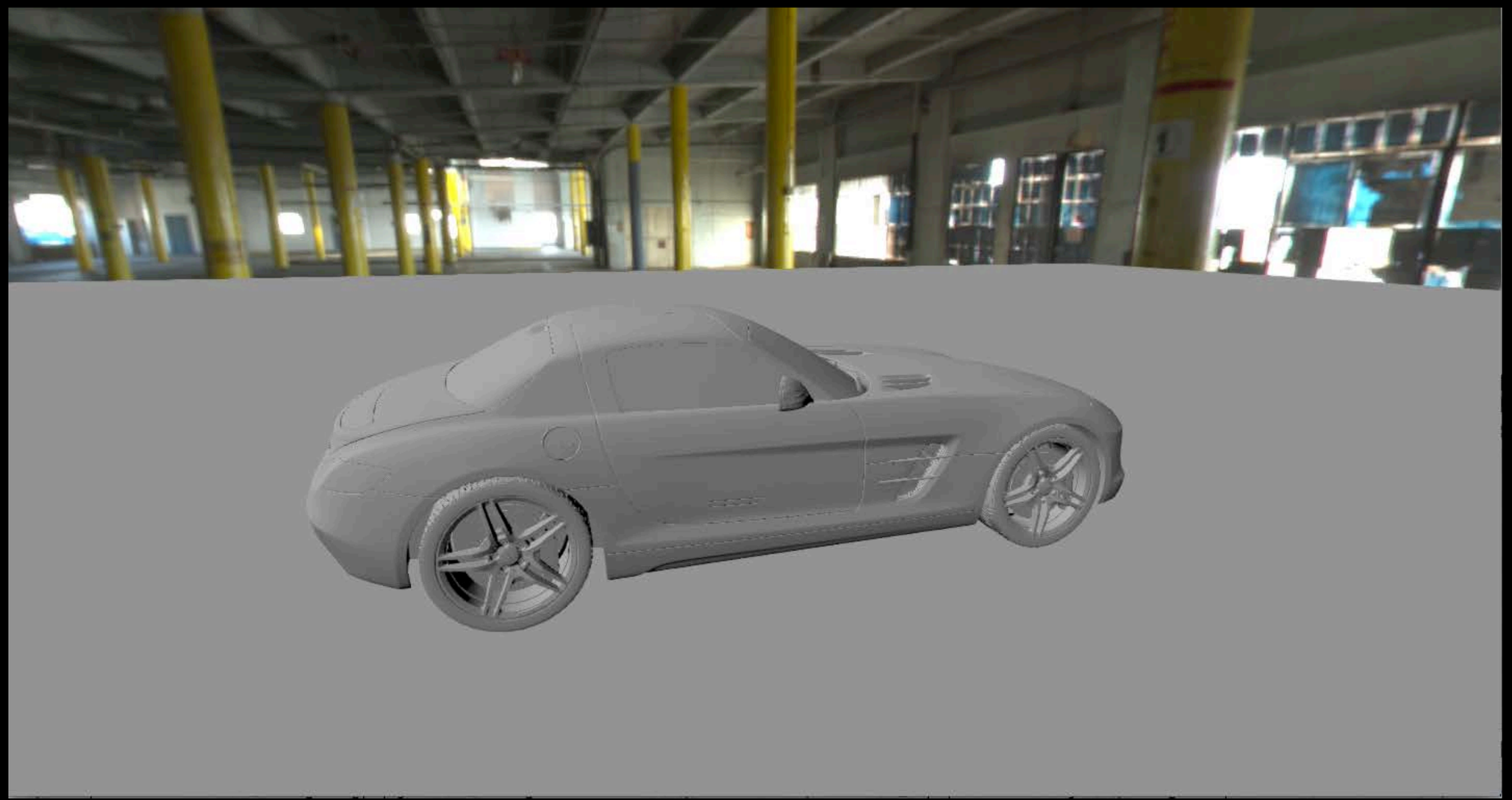
Day Five



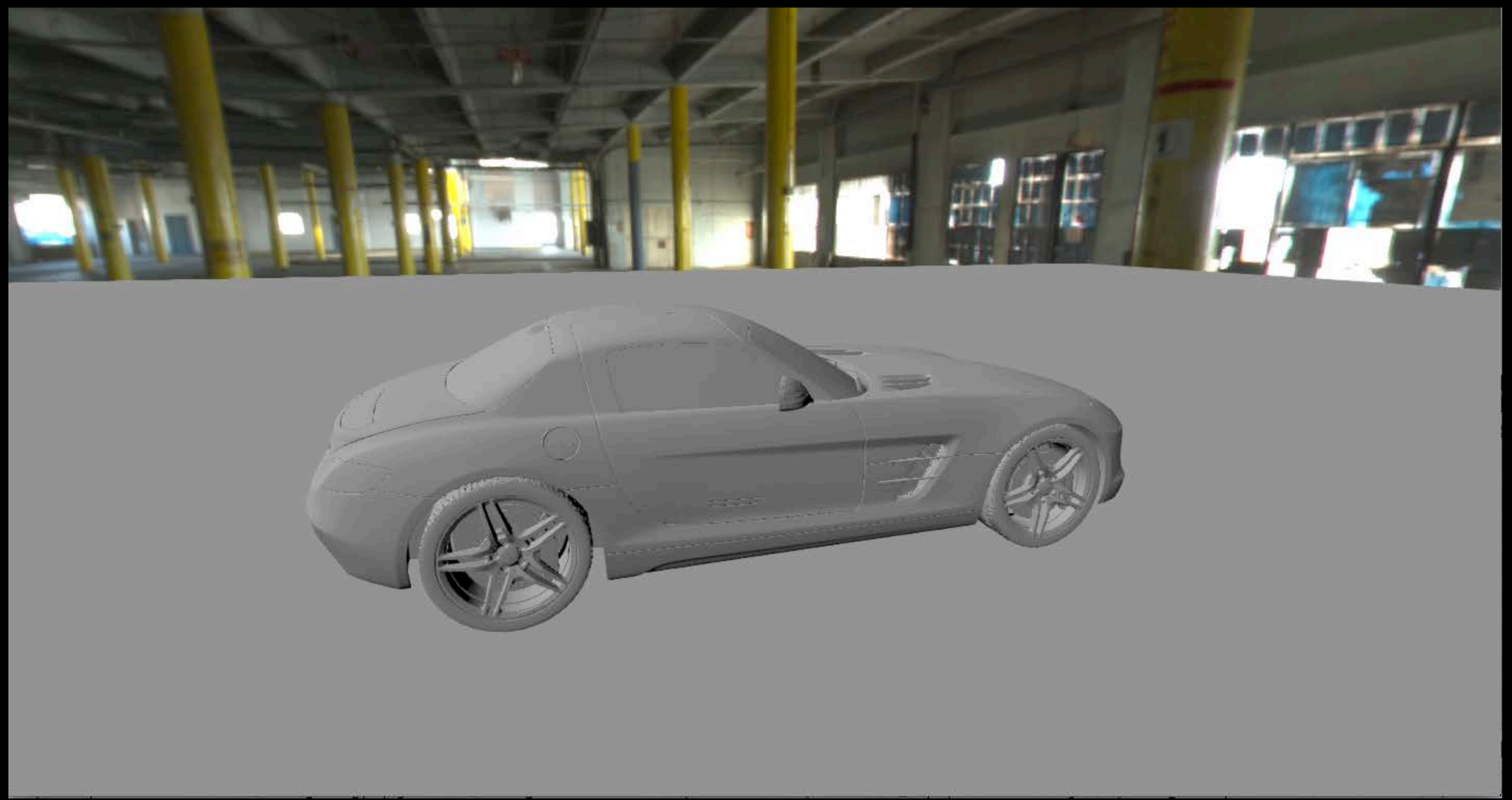
Day Five



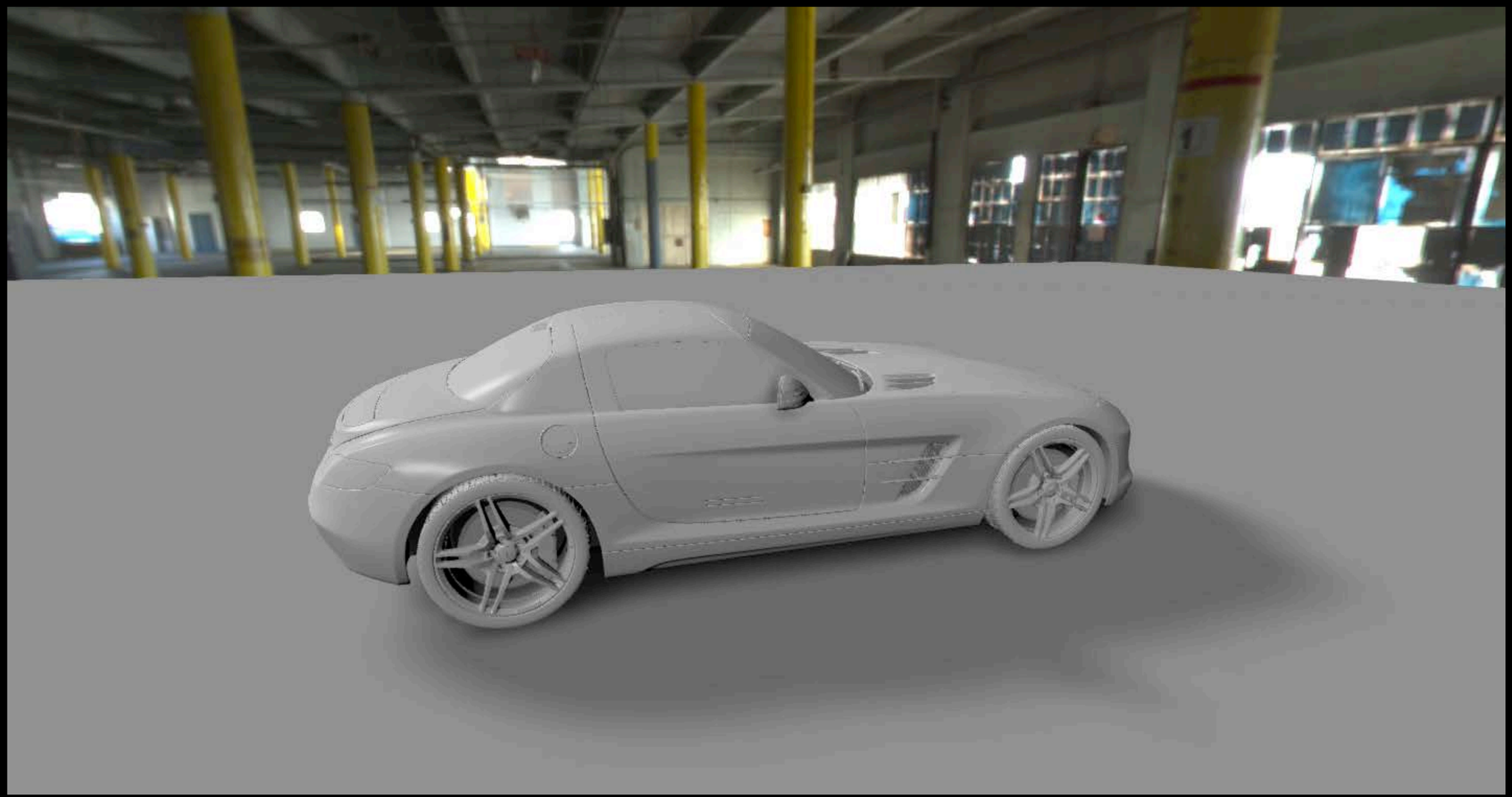
Day Fifteen



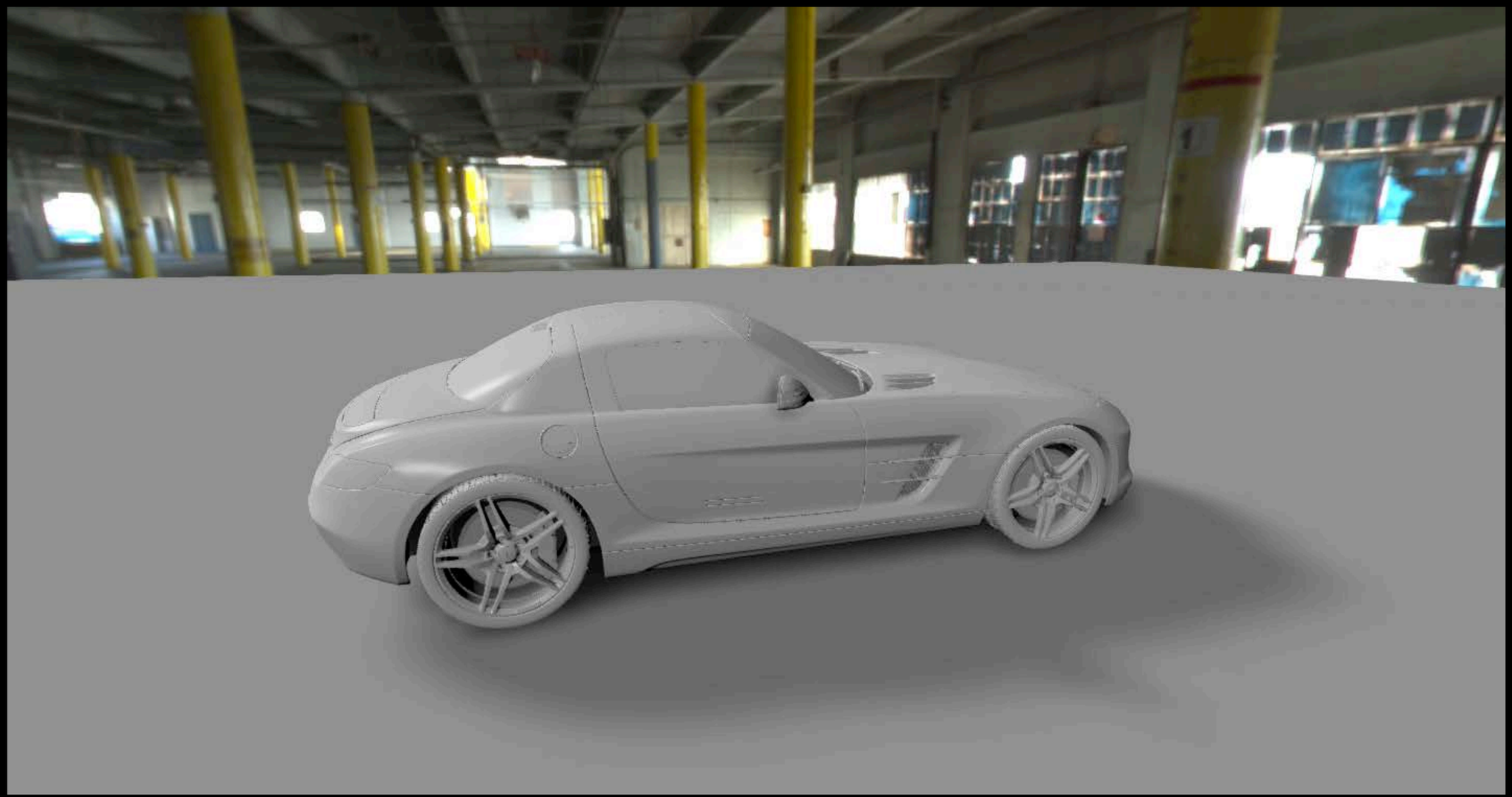
Day Fifteen



Day Twenty



Day Twenty



Day Twenty-Five

Day Twenty-Five

car_bundled.lxo* - MODO (N9 dev build)

Layouts | Model | Topology | UVEdit | Paint | Layout | Setup | Animate | Render | Scripting | Schematic Fusion | + | Only

Sculpt | Presets | Vertices | Boundary | Polygons | Materials | Items | Action Center | Symmetry | Falloff | Constrai ... | Select Through | Work Plane

Basic | Deform | Duplica... | Mesh E... | Vertex | Edge | Polygon | UV | Fusion

CocoaView

(no info)

Item L... | Shading | Groups

Filter Items | Add Item | F

Name

- car_bundled.lxo*
  - SLS (2)
  - 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 (T
    - Gradient Roughness (Tex
    - 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

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

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

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

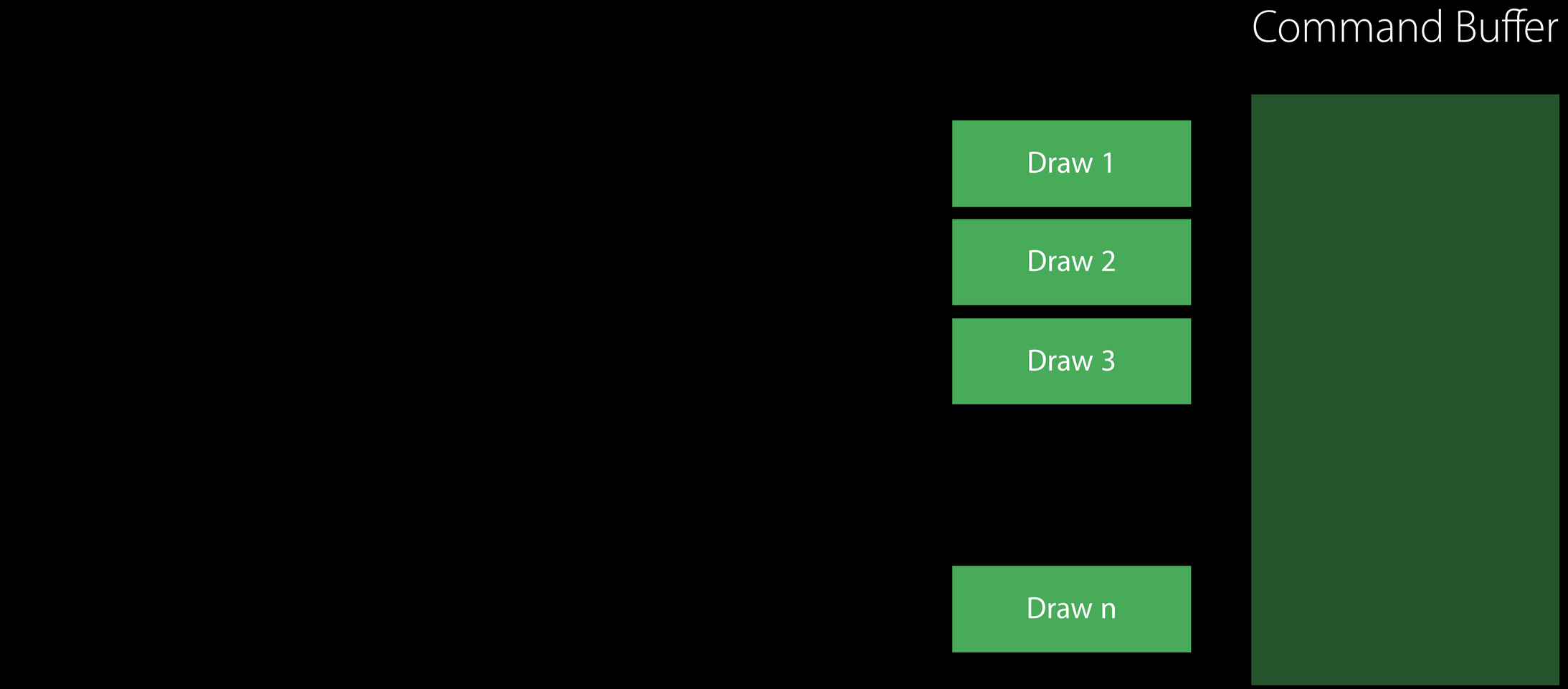
Name	Introduced	Feature Additions	Supported Devices
`iOS_GPUFamily1_v1`	iOS 8	Core Metal features for A7 devices	iPhone 5s iPad Air
`iOS_GPUFamily1_v2`	iOS 9	New texture features Multi-sample depth resolves Depth clipping support Separate stencil reference values	iPhone 6 and 6 Plus iPad Air 2
`iOS_GPUFamily2_v1`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

OS X Metal Feature Sets

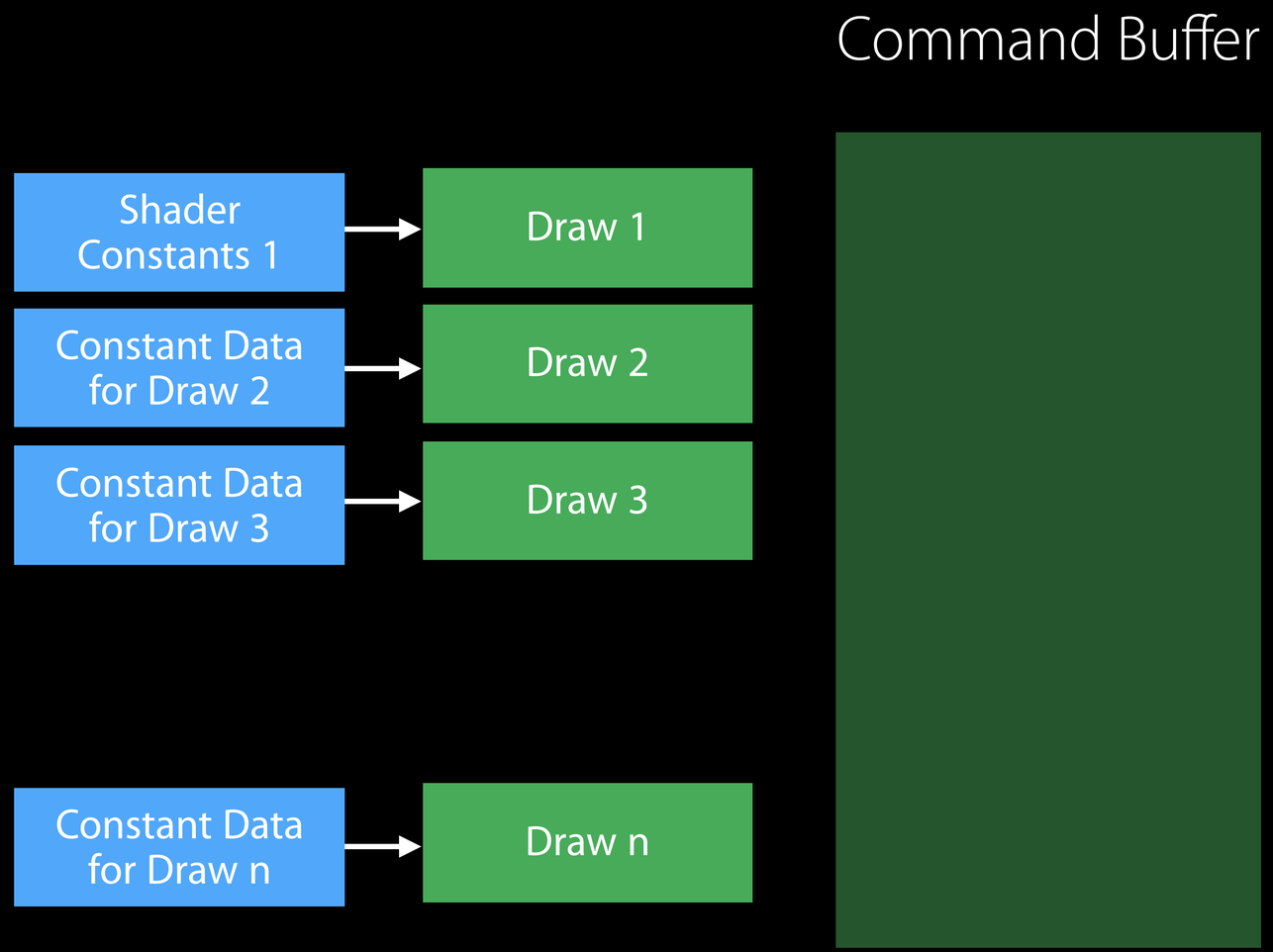
Name	Introduced	Feature Additions	Supported Devices
`OSX_GPUFamily1_v1`	OS X 10.11	Same core feature as iOS, plus BCn texture compression formats Combined depth stencil formats Managed resource model Multi-GPU device selection Draw and compute indirect counting occlusion queries Layer select Texture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

Shader Constant Updates

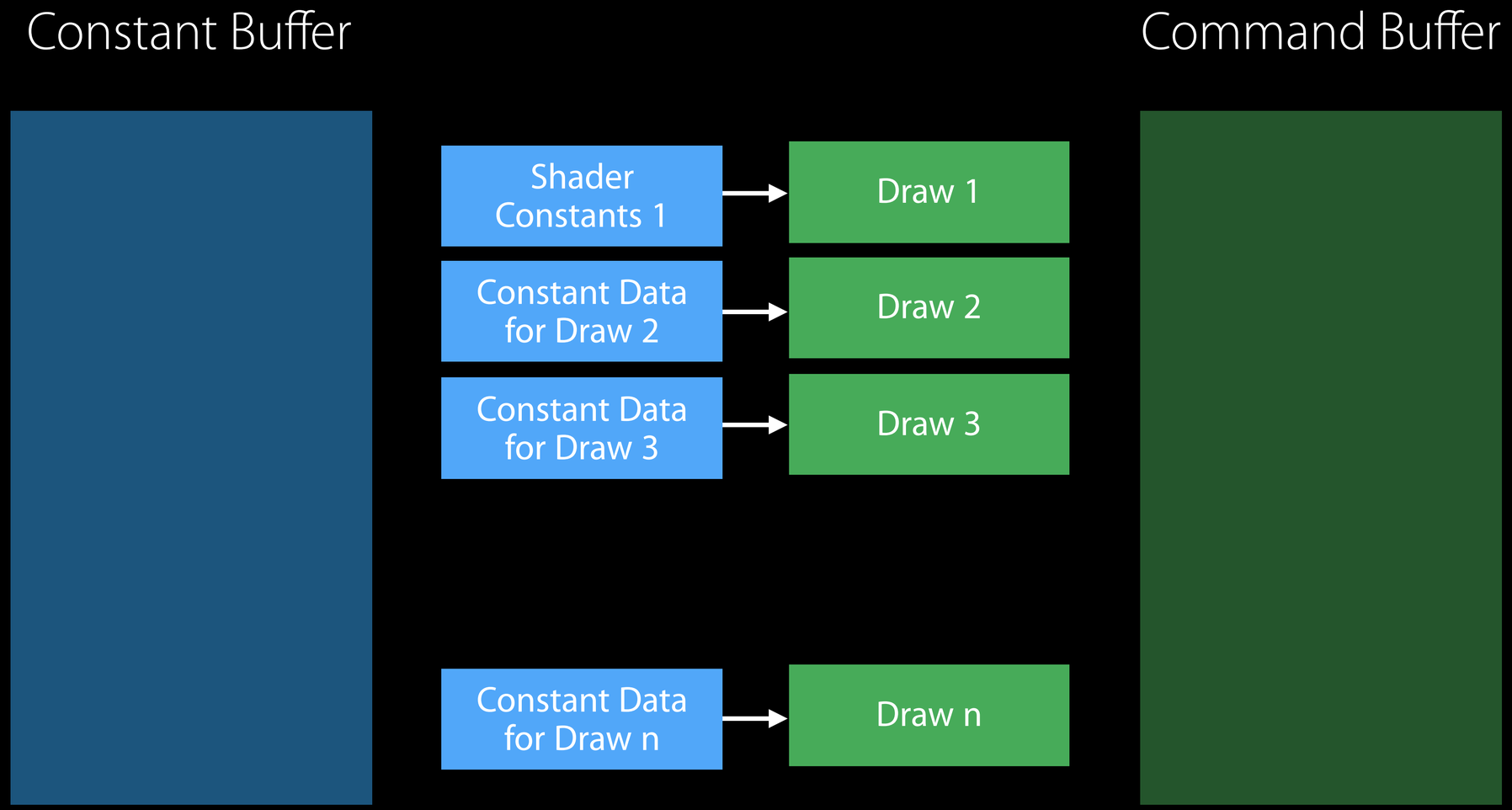
Shader Constant Updates



Shader Constant Updates



Shader Constant Updates



Shader Constant Updates

Constant Buffer

Shader Constants 1

Constant Data for Draw 2

Constant Data for Draw 3

Constant Data for Draw n

Command Buffer

Draw 1

Draw 2

Draw 3

Draw n

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

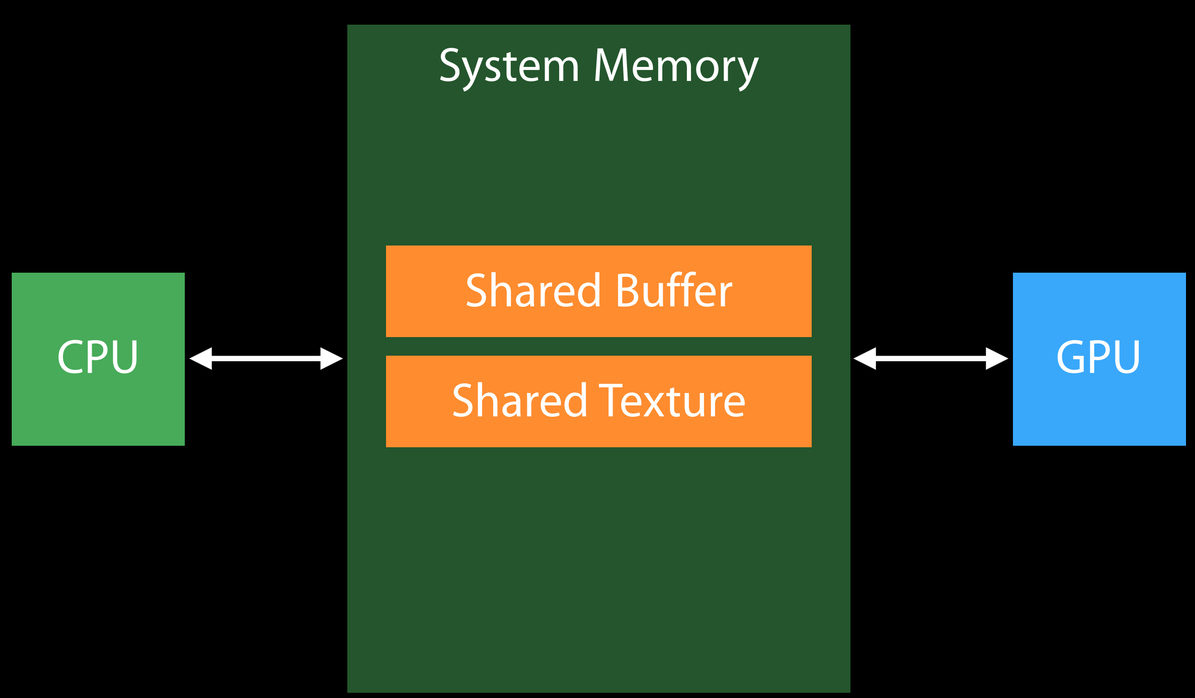
New storage modes to specify where the resource should reside

- 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

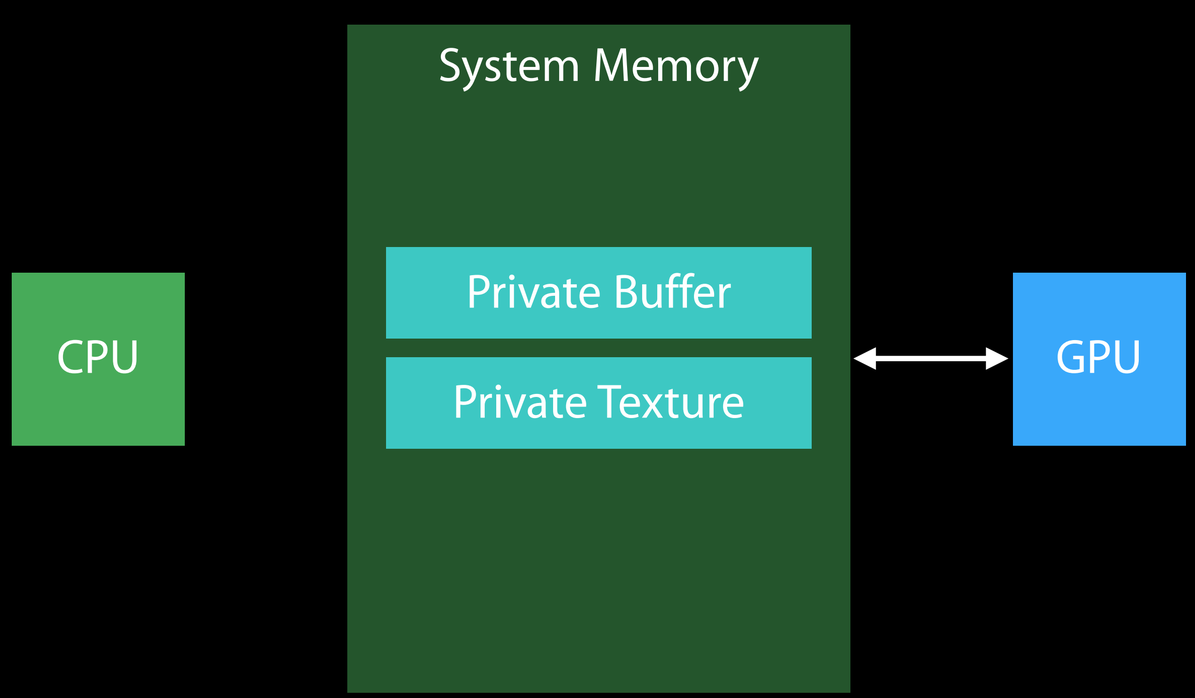


Private Storage Mode

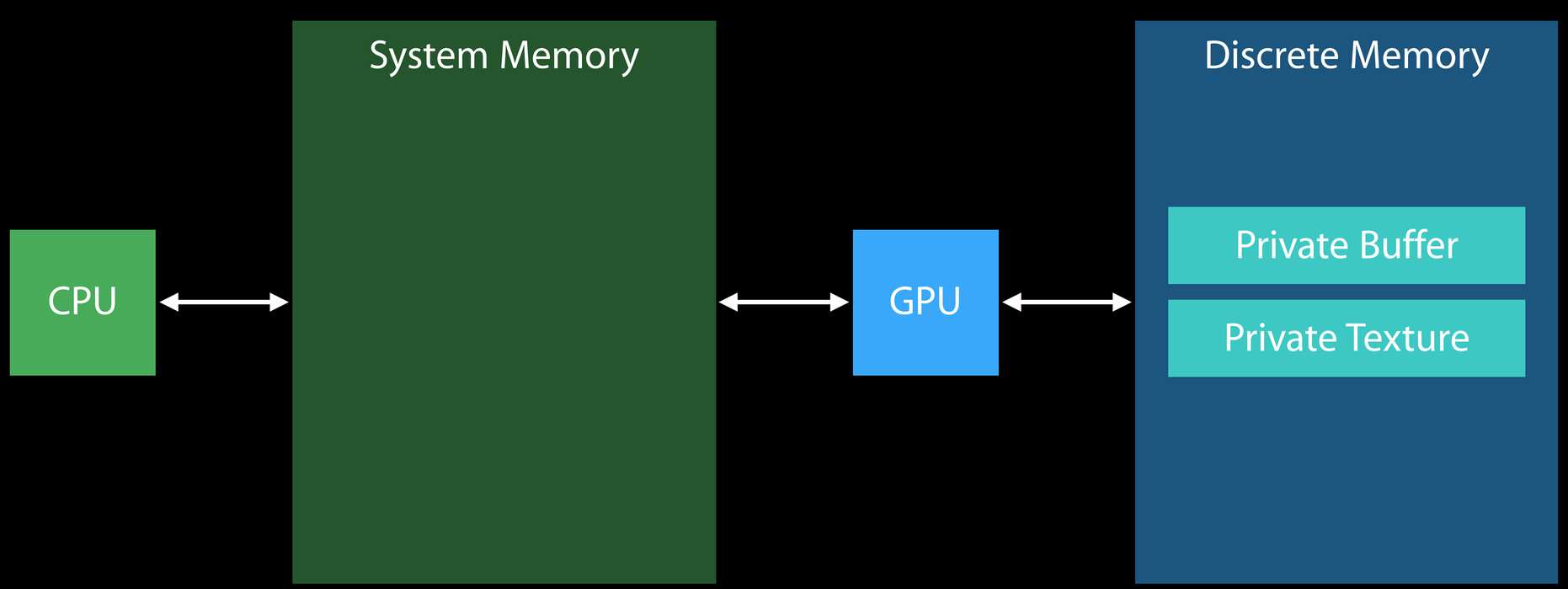
New with iOS 9 and OS X

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

Metal can store data more optimally for the GPU



Private Storage Mode with Discrete Memory

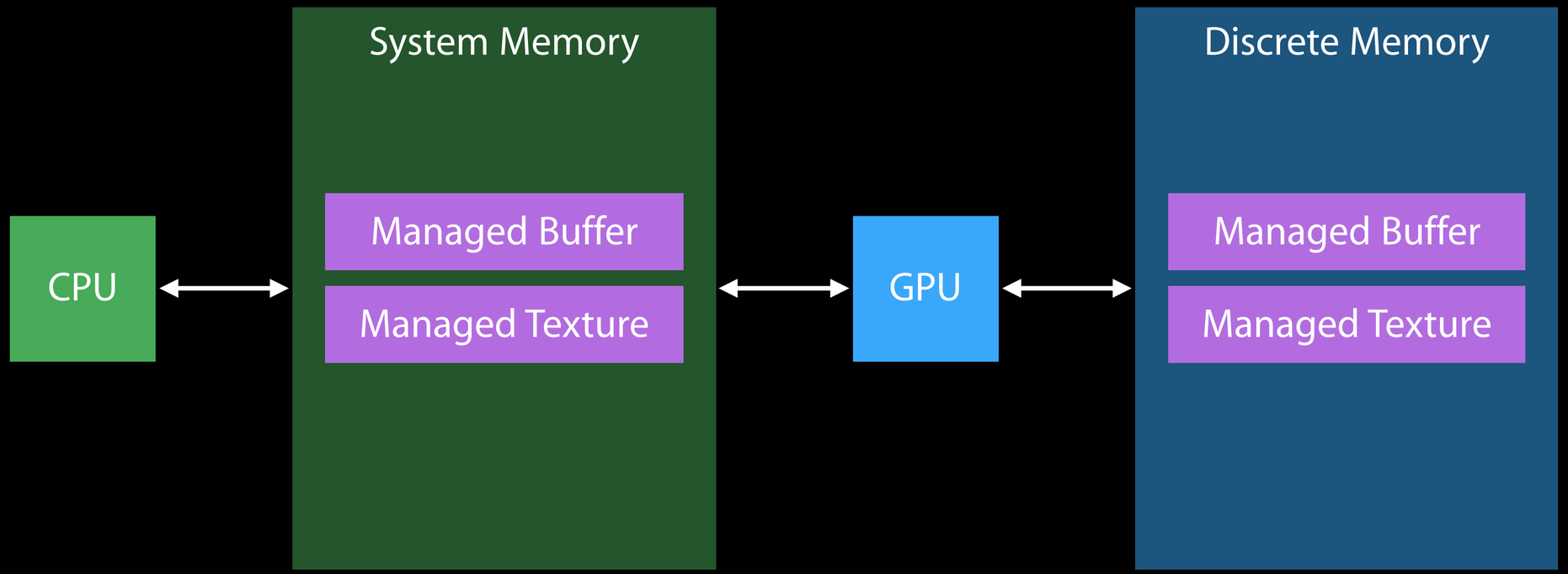


Managed Storage Mode

New with OS X

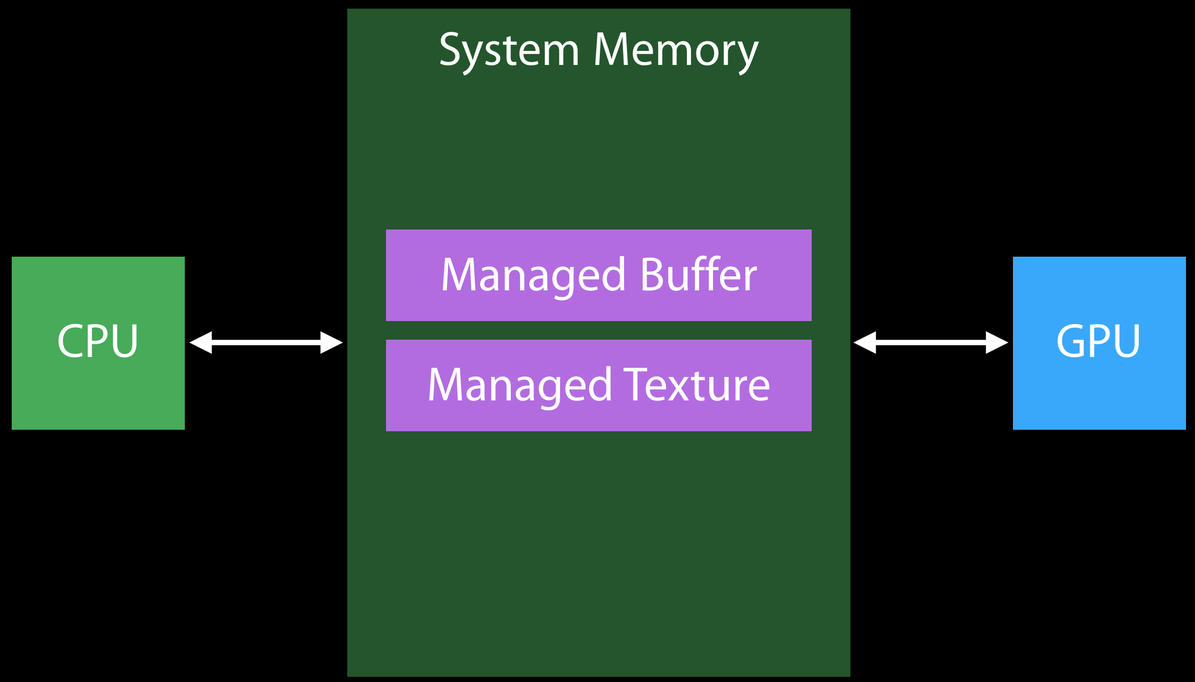
Metal manages where the resource resides

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:NSRangeMake(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:NSRangeMake(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:NSRangeMake(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

- NSViewGlobalFrameDidChangeNotification

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

- NSViewGlobalFrameDidChangeNotification

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

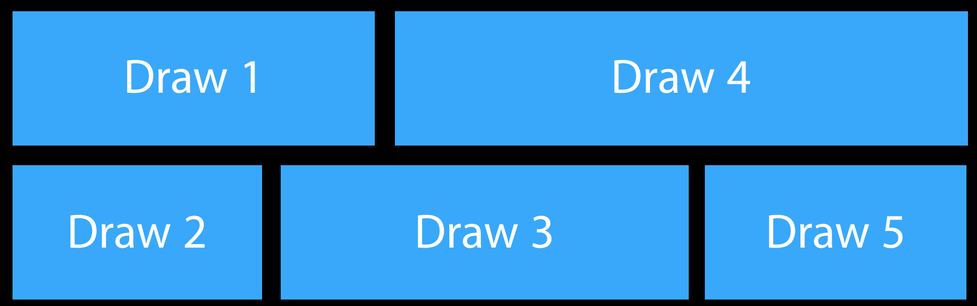
```
struct VSOut {
    float4 position [[position]];
    ushort layer [[render_target_array_index]];
}
```

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

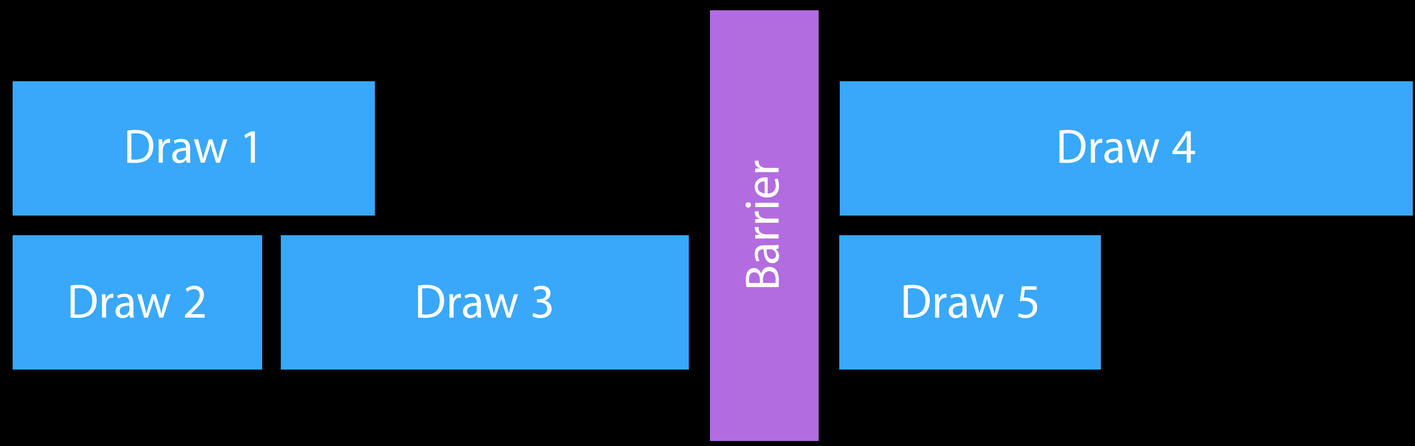


Texture Barriers

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

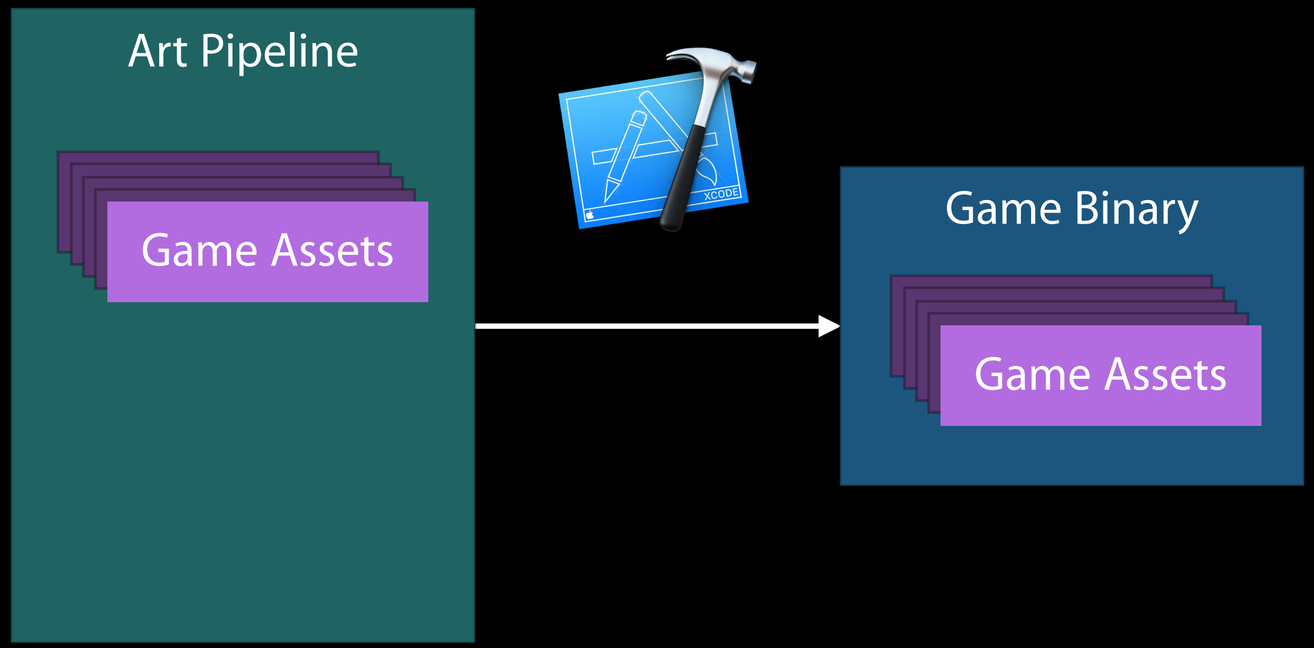
New Texture Features

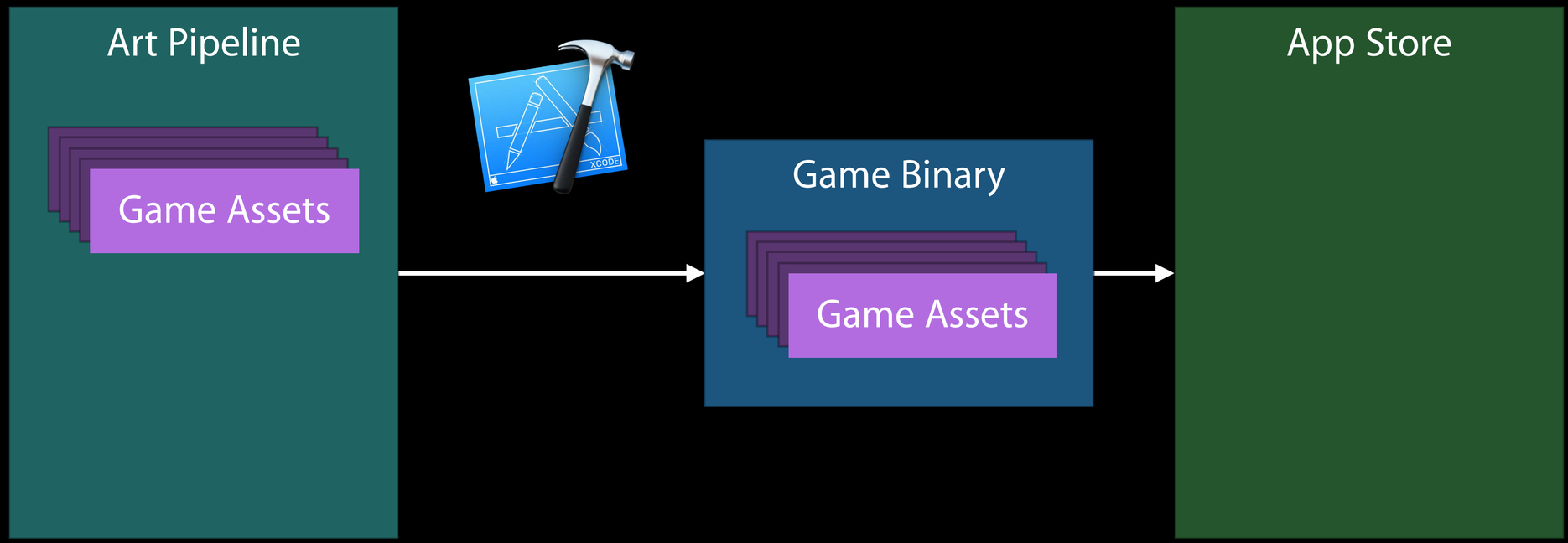
OS X texture compression formats

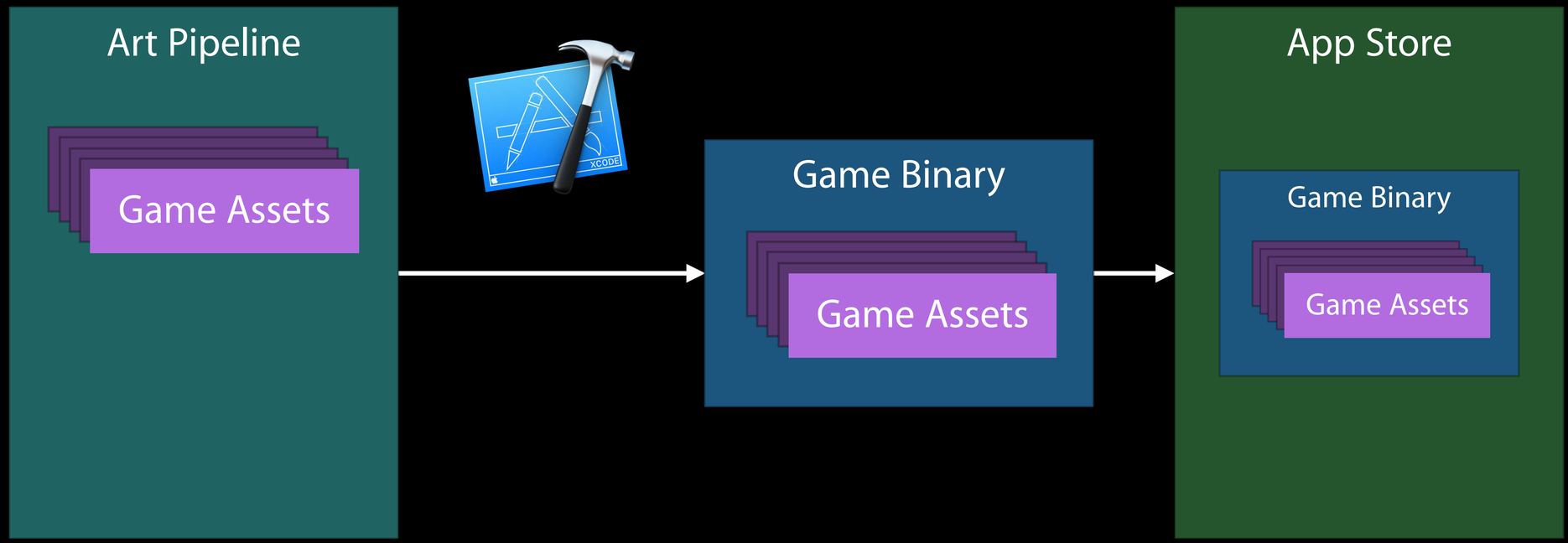
Format	Bits Per Pixel	Also Known As	Properties
BC1	4	S3TC, DXT1	RGB content Very fast encoding
BC2, BC3	8	S3TC, DXT3, DXT5	RGBA content Very fast encoding
BC4, BC5	4 - One channel 8 - Two channels	RGTC	Height/Bump Maps Normal Maps Alpha Masks
BC6, BC7	8	BPTC	RGBA content Slowest encoding

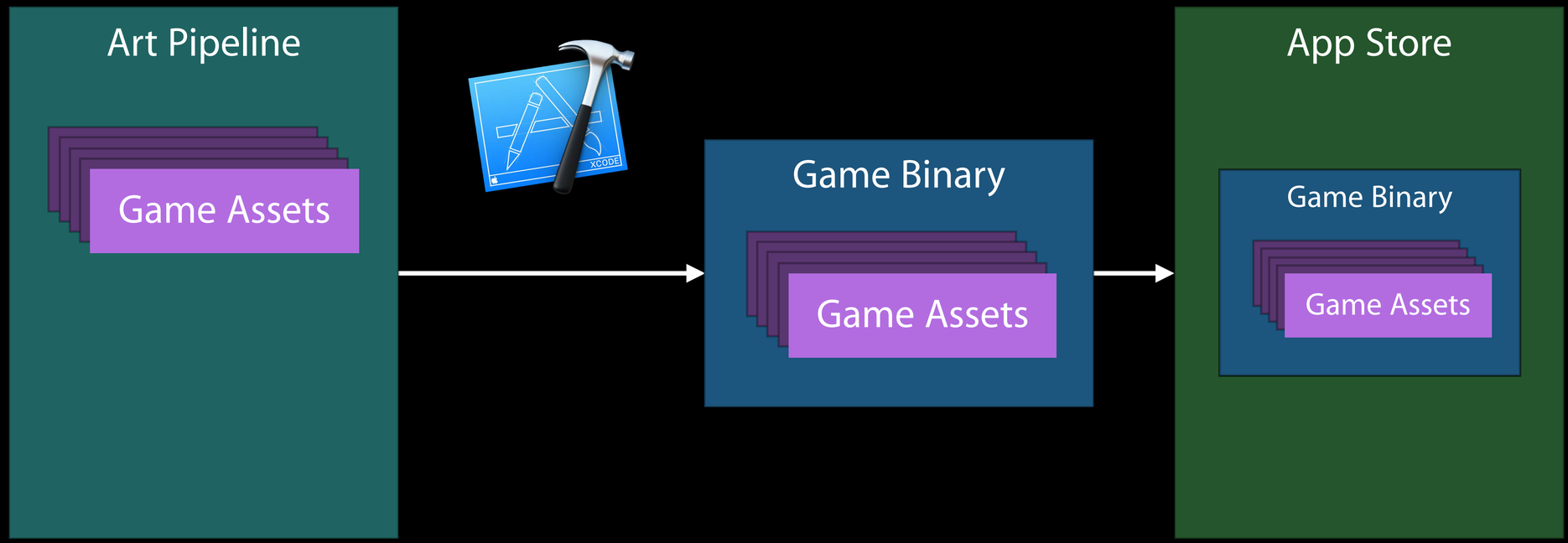
Metal and App Thinning


Art Pipeline
Game Assets


Art Pipeline
Game Assets
XCODE
Game Binary
Game Assets

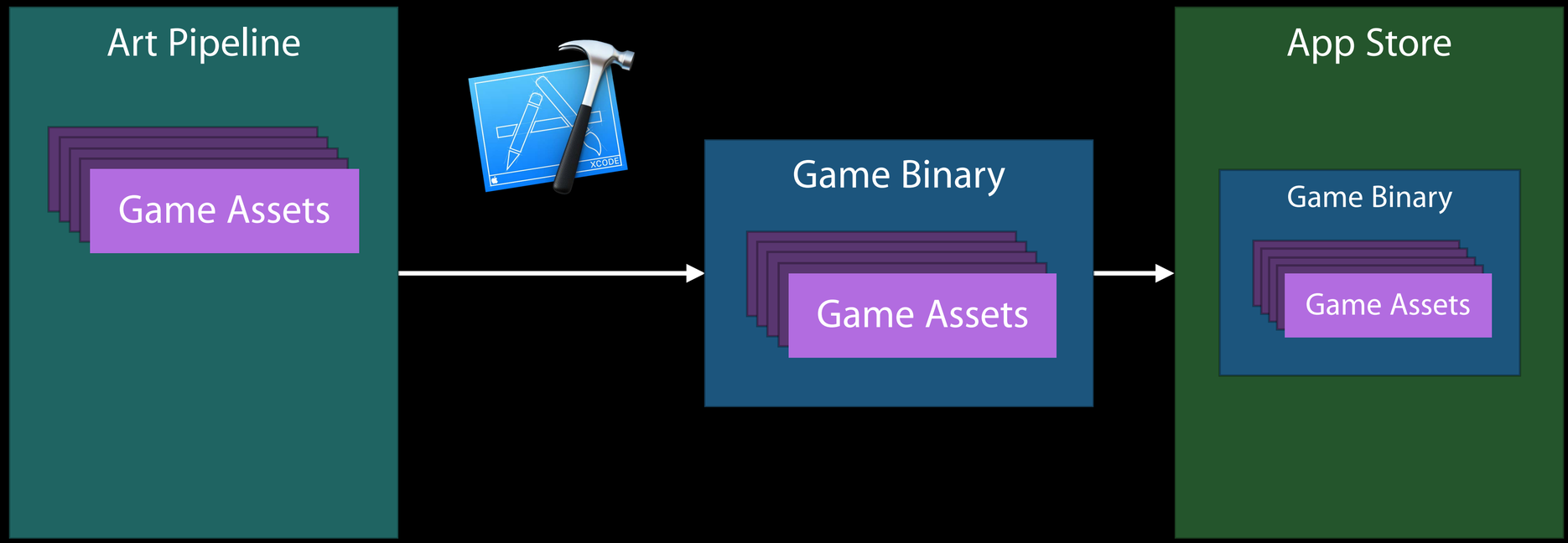

Art Pipeline
Game Assets
XCODE
Game Binary
Game Assets
App Store

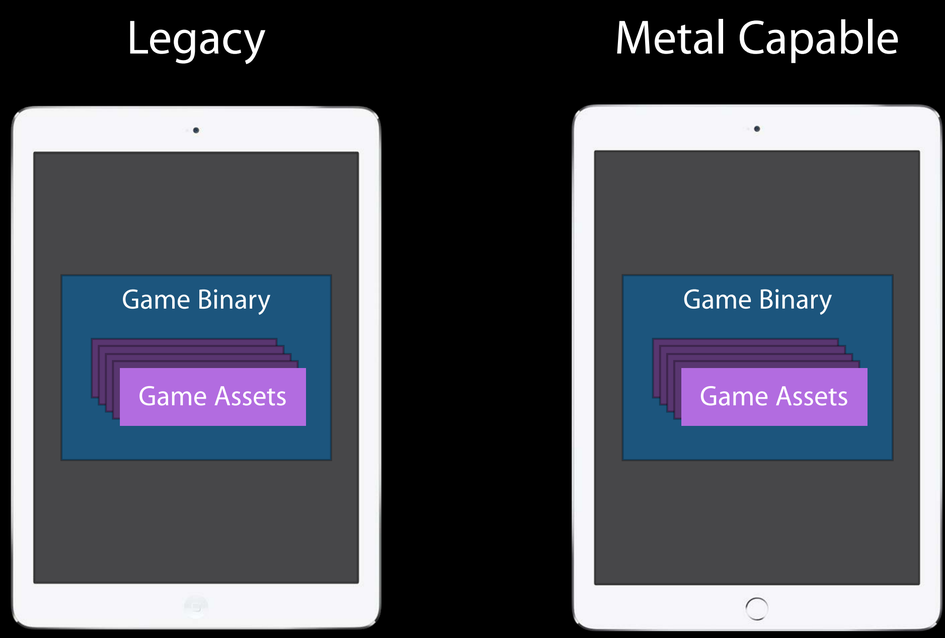

Art Pipeline
Game Assets
XCODE
Game Binary
Game Assets
App Store
Game Binary
Game Assets

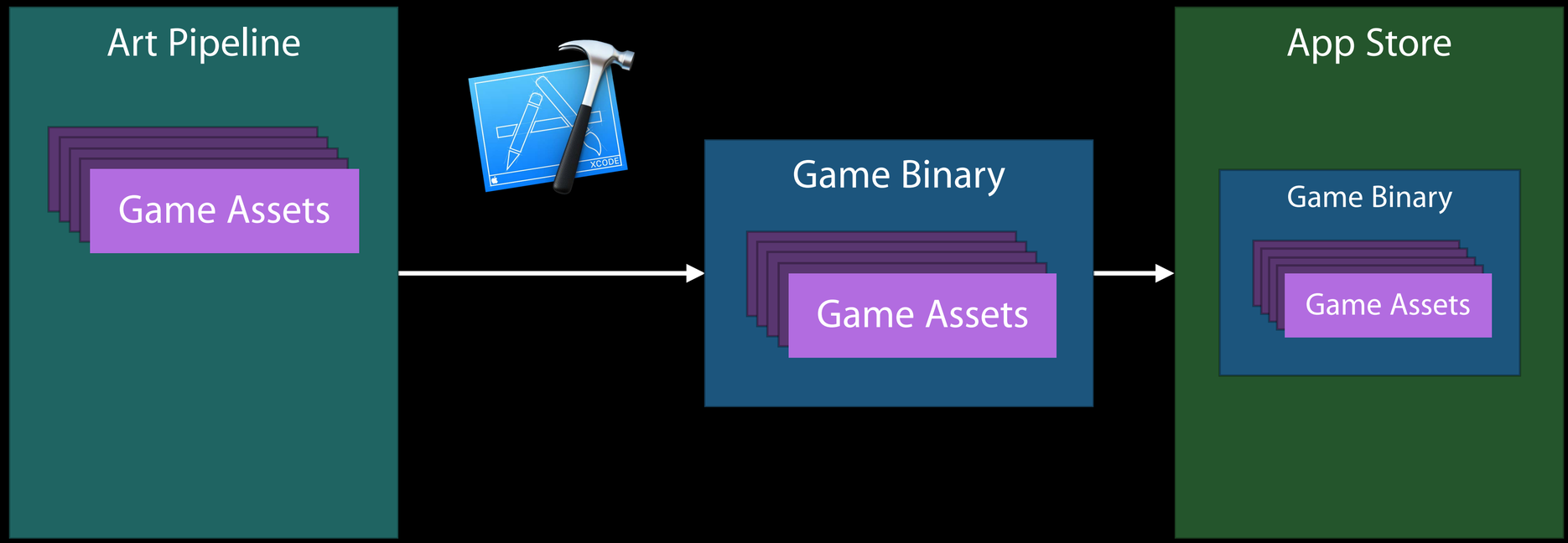


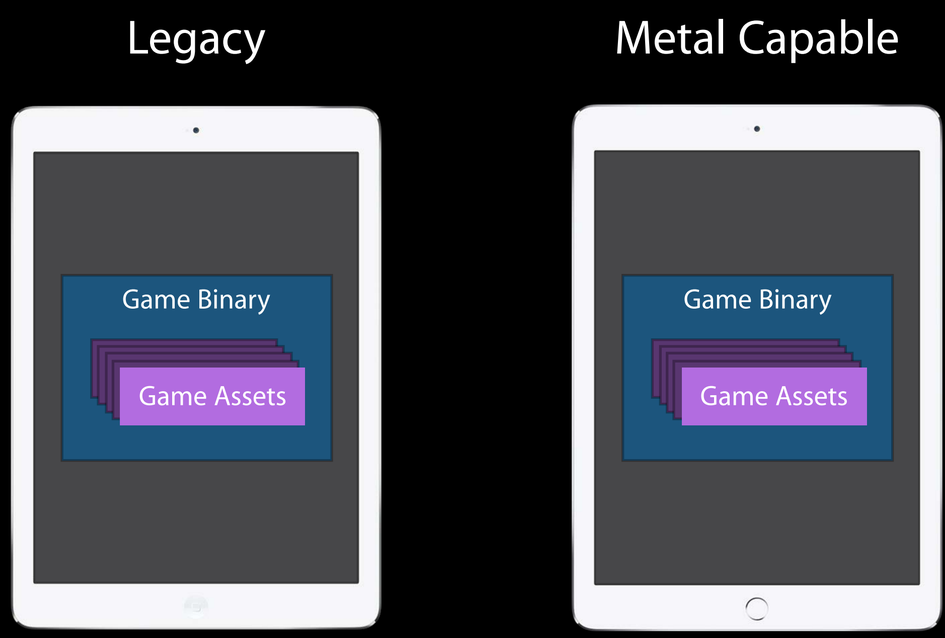
Legacy

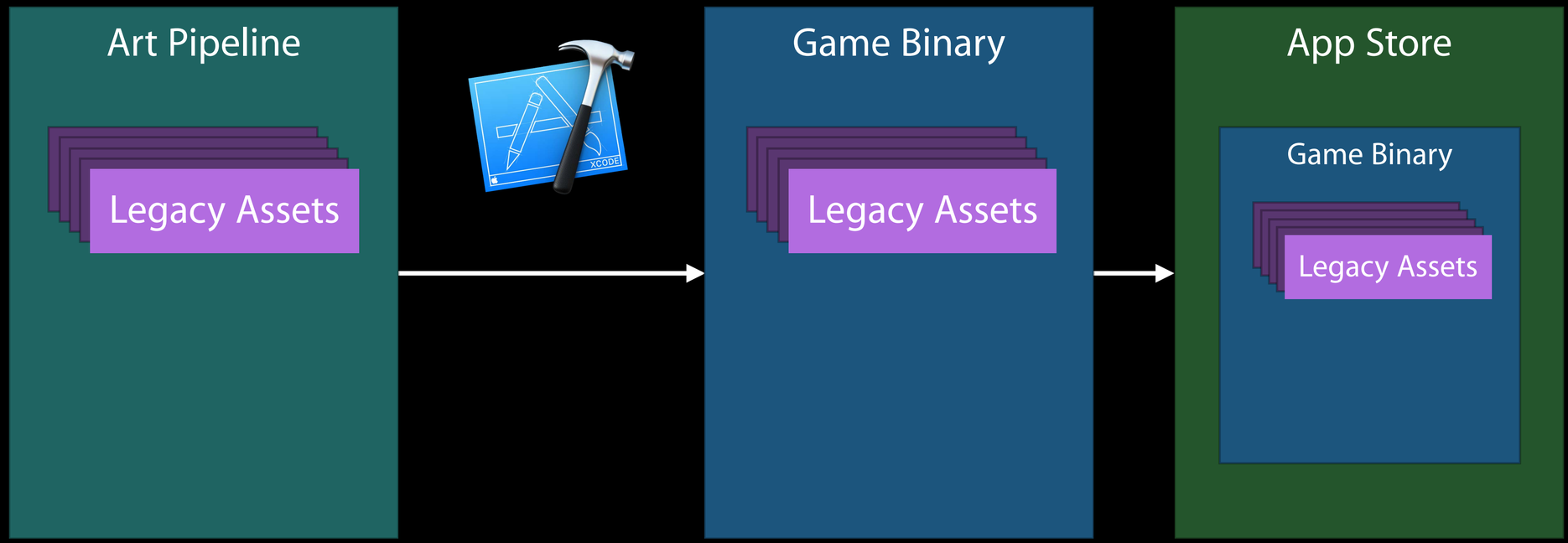
Metal Capable

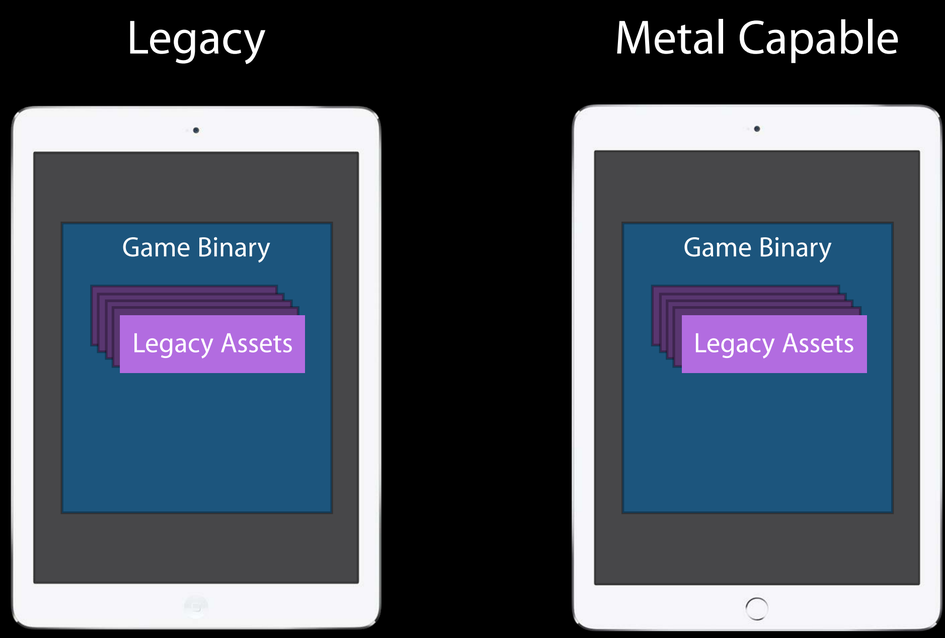

Art Pipeline
Game Assets
XCODE
Game Binary
Game Assets
App Store
Game Binary
Game Assets

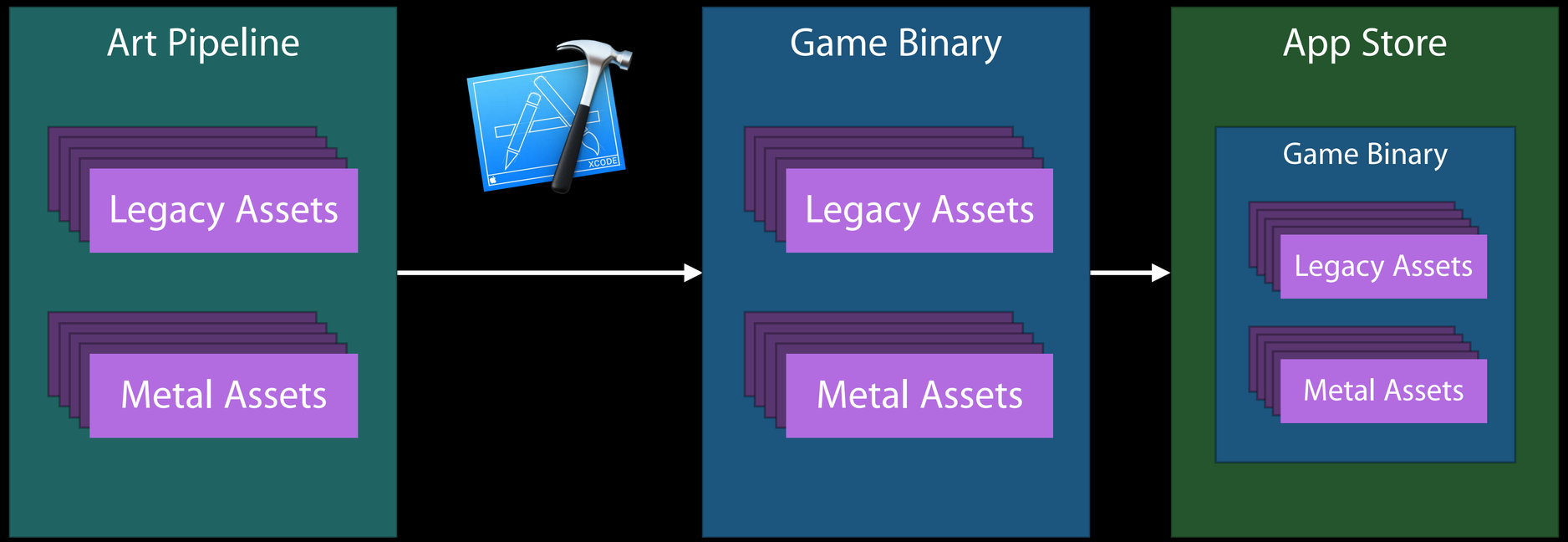

Legacy
Game Binary
Game Assets
Metal Capable
Game Binary
Game Assets

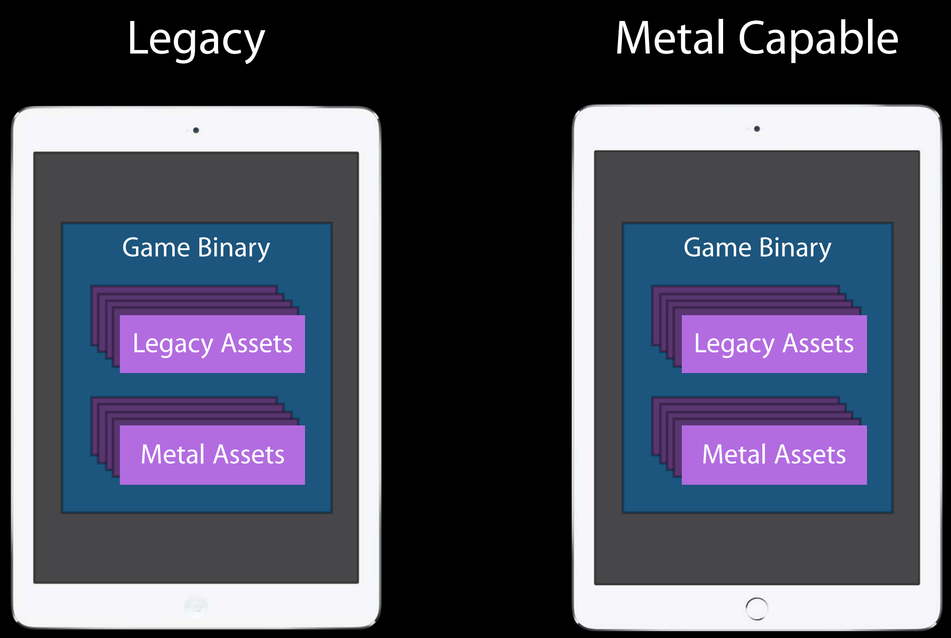

Art Pipeline
Game Assets
XCODE
Game Binary
Game Assets
App Store
Game Binary
Game Assets

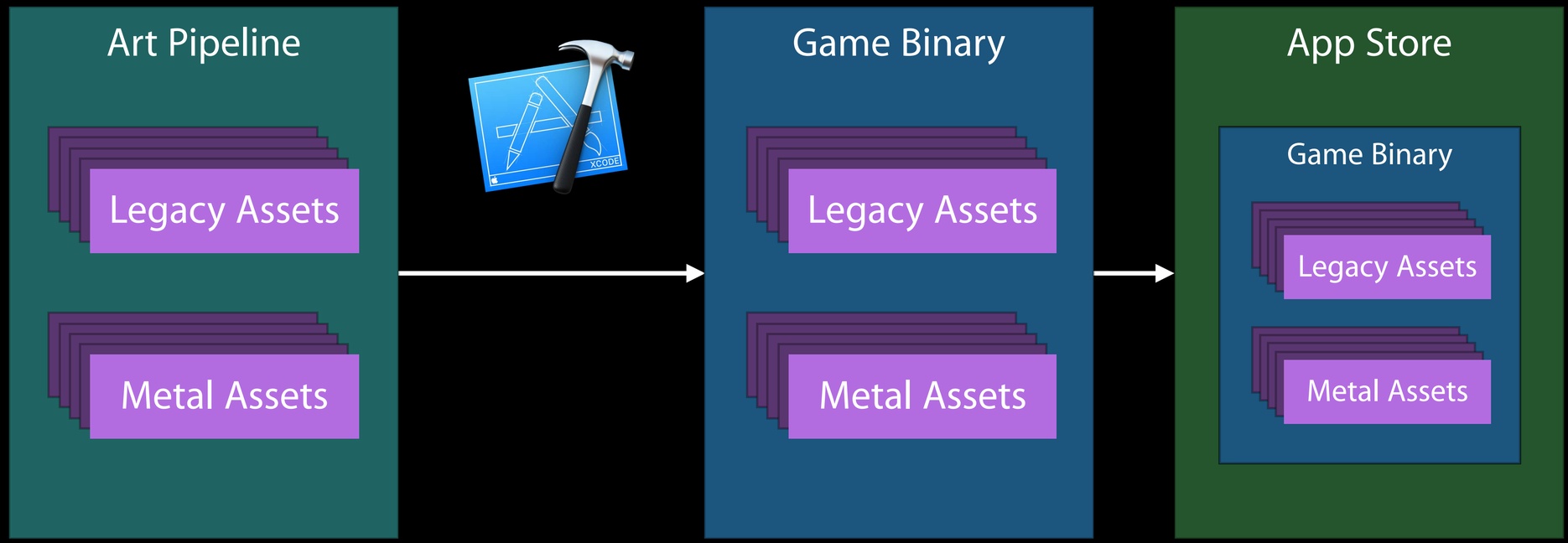

Legacy
Game Binary
Game Assets
Metal Capable
Game Binary
Game Assets

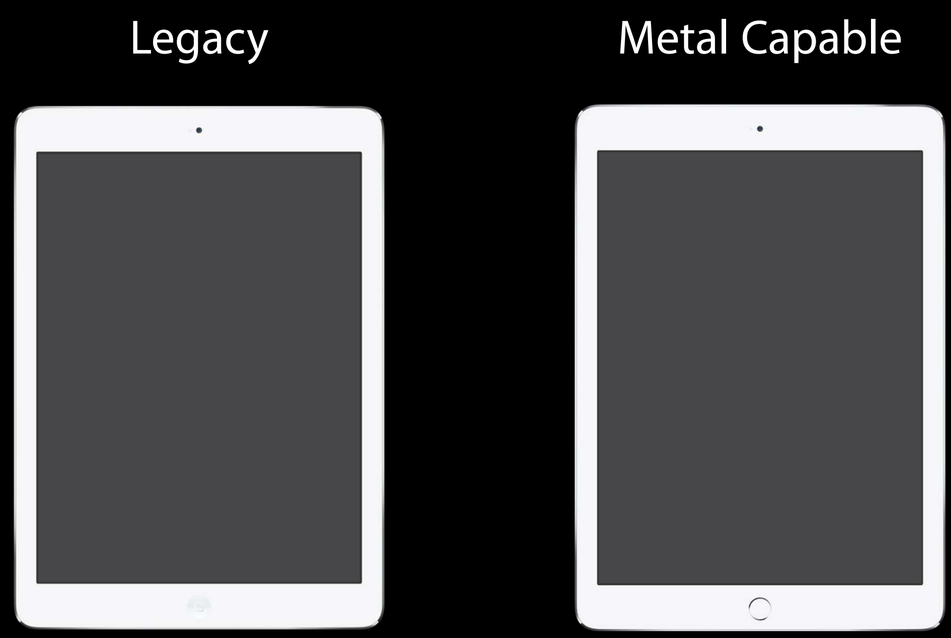

Art Pipeline
Legacy Assets
XCODE
Game Binary
Legacy Assets
App Store
Game Binary
Legacy Assets

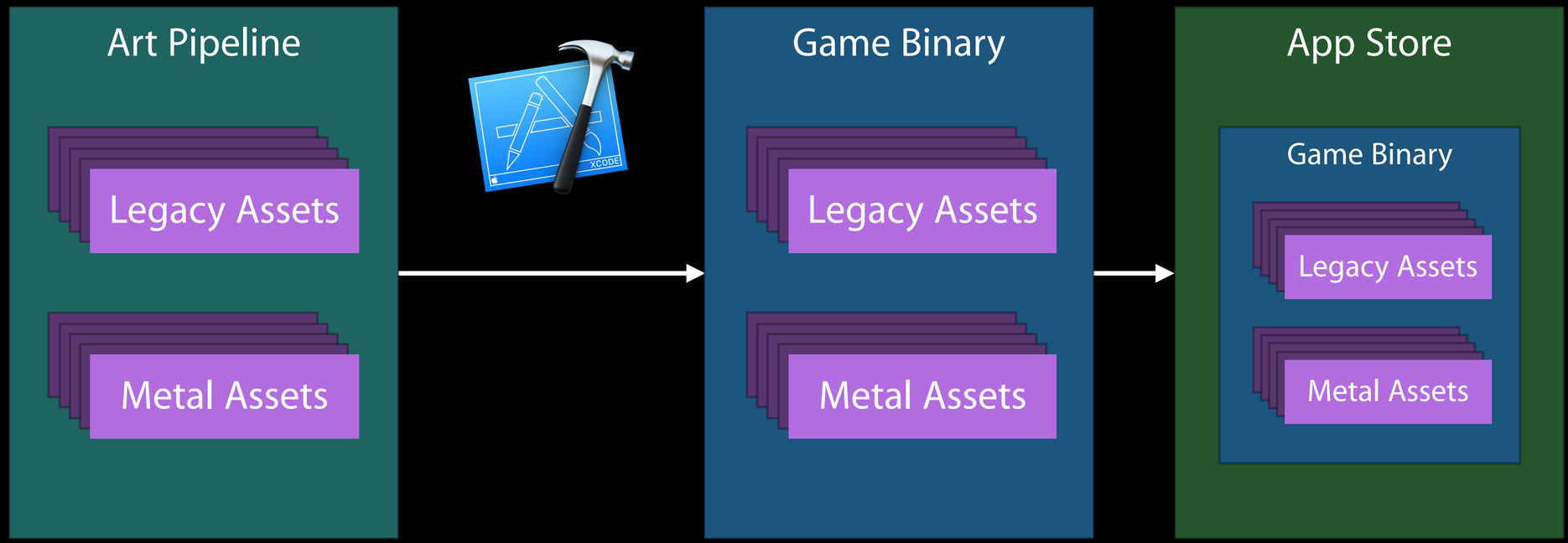

Legacy
Game Binary
Legacy Assets
Metal Capable
Game Binary
Legacy Assets

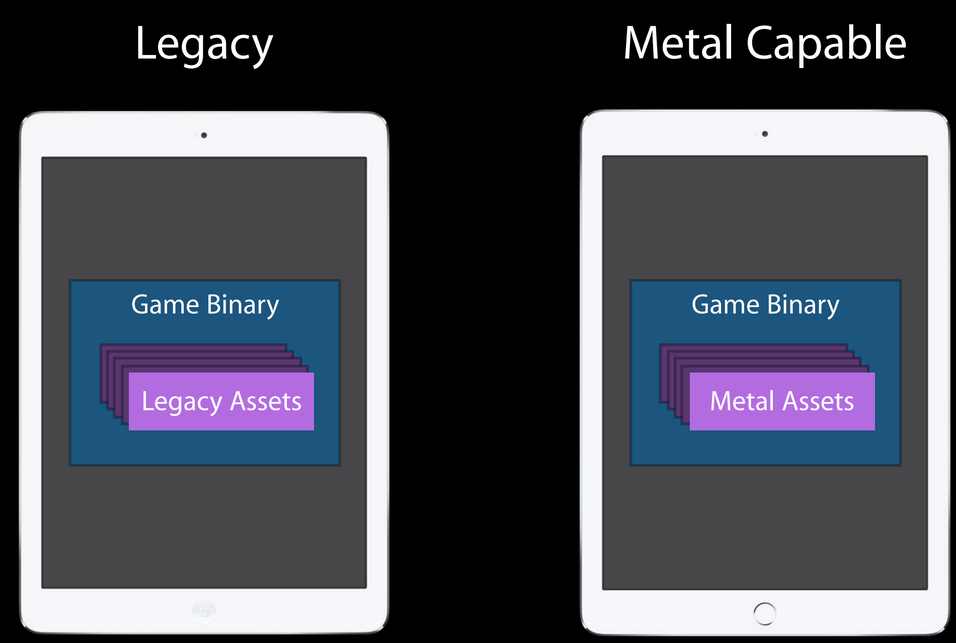

Art Pipeline
Legacy Assets
Metal Assets
XCODE
Game Binary
Legacy Assets
Metal Assets
App Store
Game Binary
Legacy Assets
Metal Assets

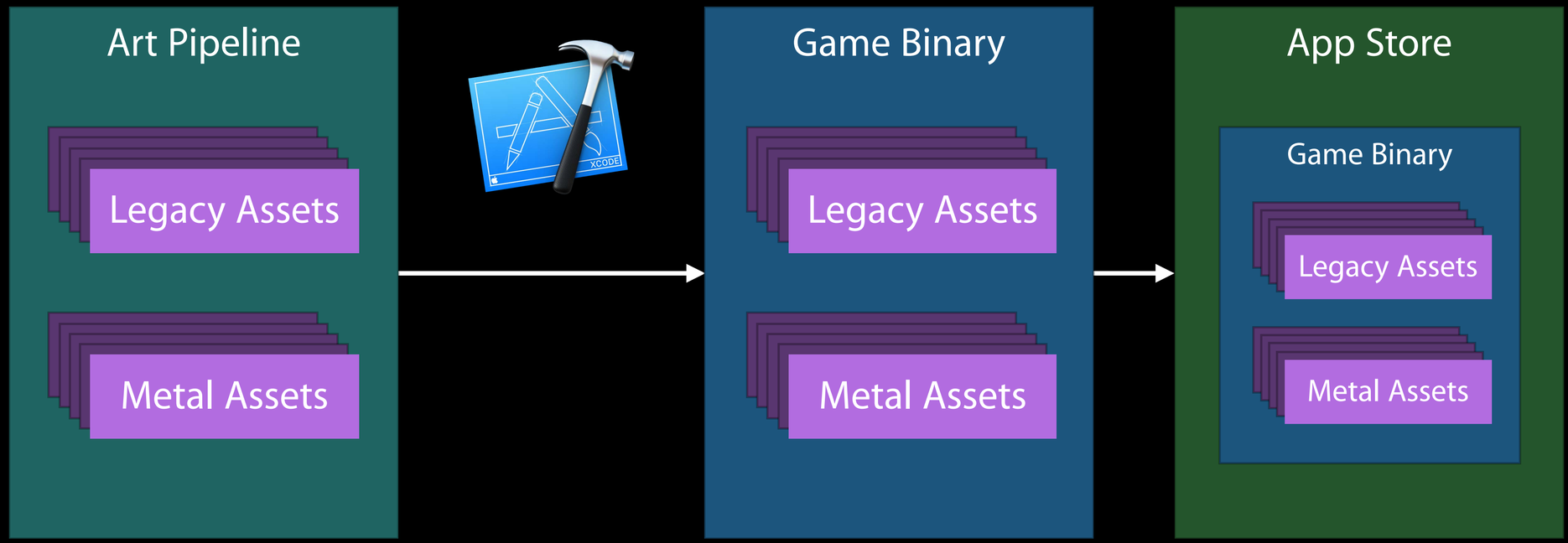

Legacy
Game Binary
Legacy Assets
Metal Assets
Metal Capable
Game Binary
Legacy Assets
Metal Assets


Art Pipeline
Legacy Assets
Metal Assets
XCODE
Game Binary
Legacy Assets
Metal Assets
App Store
Game Binary
Legacy Assets
Metal Assets

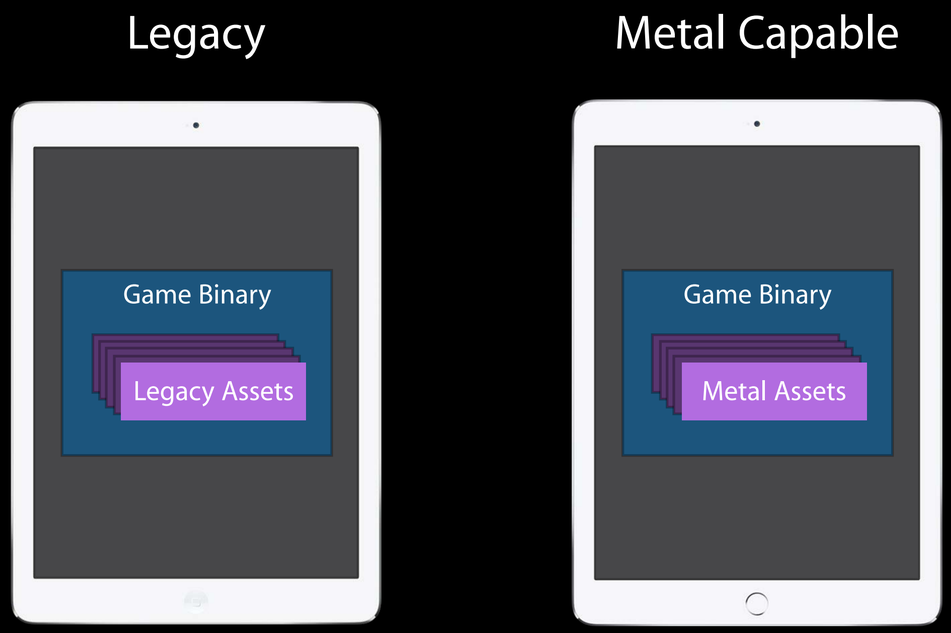

Legacy
Metal Capable


Art Pipeline
Legacy Assets
Metal Assets
XCODE
Game Binary
Legacy Assets
Metal Assets
App Store
Game Binary
Legacy Assets
Metal Assets


Legacy
Game Binary
Legacy Assets
Metal Capable
Game Binary
Metal Assets



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
Metal GPUFamily2			
Metal GPUFamily1			
OpenGL ES Legacy			

256x256 RG8

512x512 RG8

512x512 EAC

512x512 ASTC

1024x1024 ASTC

Capability Matrix

Extended normal map example

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

256x256 RG8

512x512 RG8

512x512 EAC

512x512 ASTC

Capability Matrix

Extended normal map example

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

256x256 RG8

512x512 RG8

512x512 EAC

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

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	

T...a 〉 iPhone 6 | TestGPUData | Build TestGPUData: **Succeeded** | Today at 4:13 PM

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

Show Slicing

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

T...a 〉 iPhone 6 | TestGPUData | Build TestGPUData: **Succeeded** | Today at 4:13 PM

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

Show Slicing

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

T...a 〉 iPhone 6 | TestGPUData | Build TestGPUData: **Succeeded** | Today at 4:13 PM

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

Show Slicing

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

T...a 〉 iPhone 6 | TestGPUData | Build TestGPUData: **Succeeded** | Today at 4:13 PM

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

Universal

Show Slicing

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

Custom Tools Pipelines

Custom Tools Pipelines

Publicly documented JSON file format

Easily integrated into custom toolchain

Contents.json

```
{
    "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

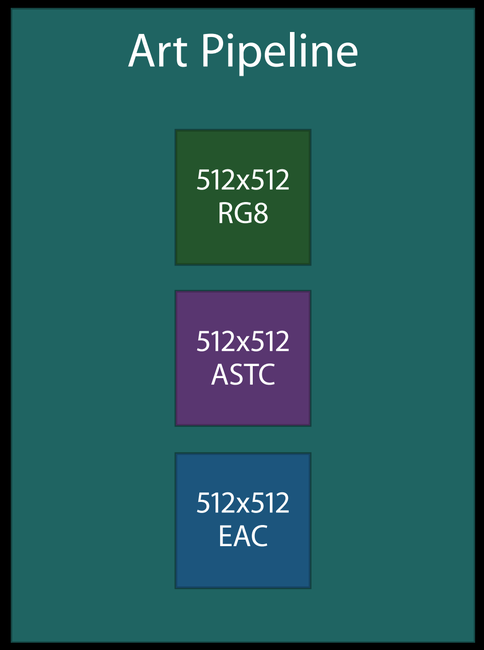
NSDataAsset class provides correctly matched data resource from Asset Catalog

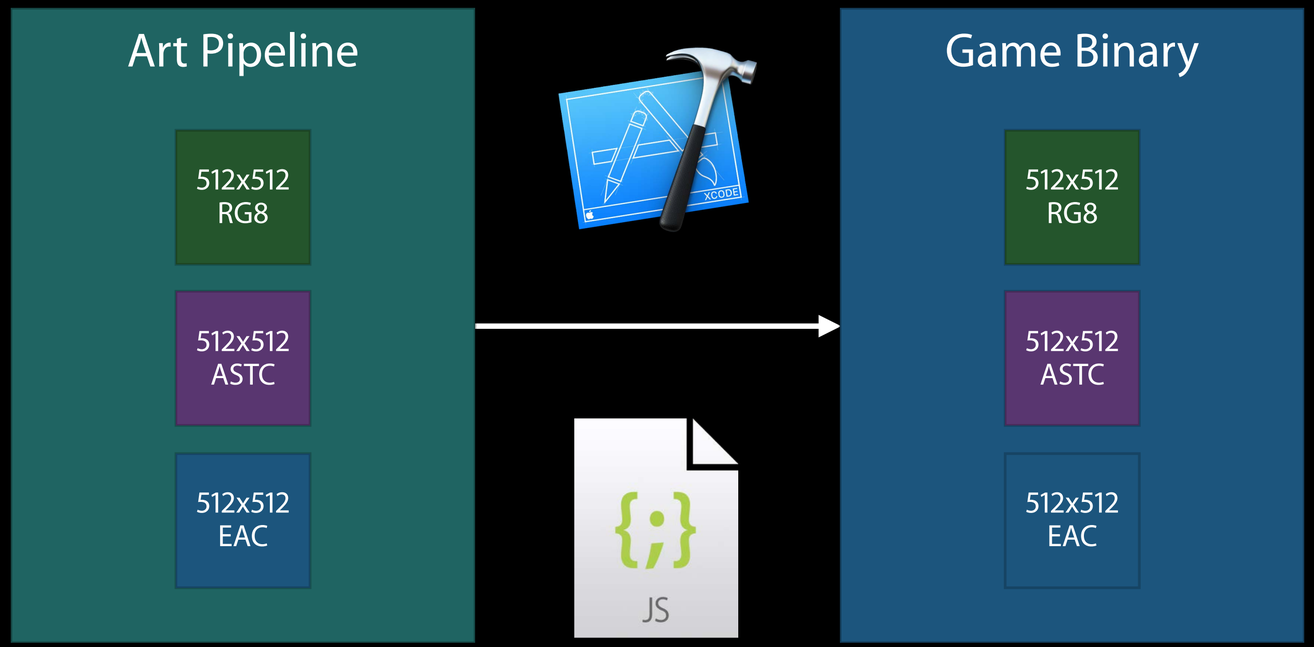
```
#import <UIKIt/NSDataAsset.h>
```

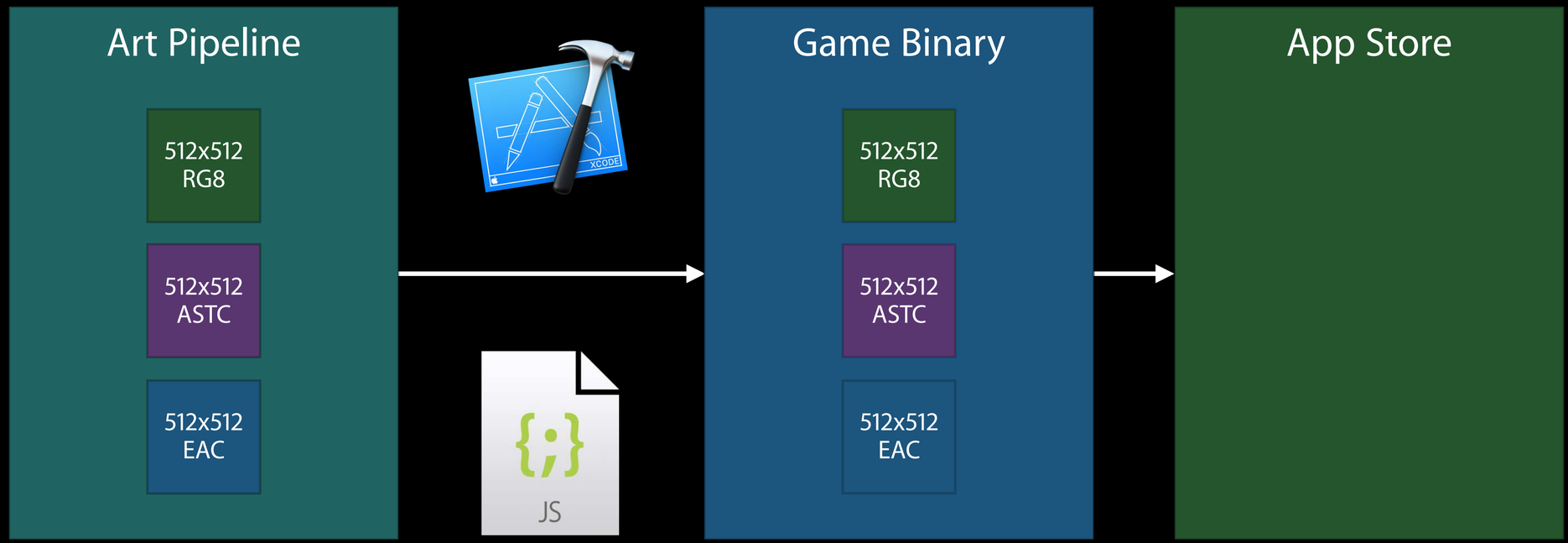
Retrieving Named Data

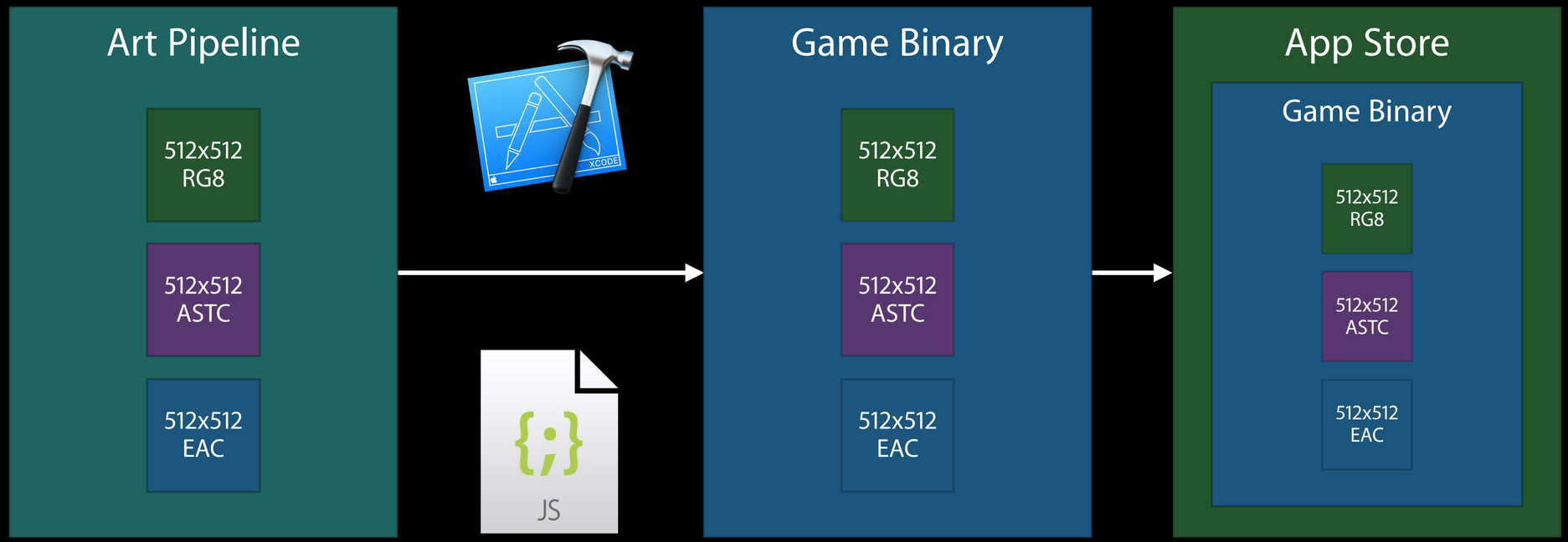
NSDataAsset class provides correctly matched data resource from Asset Catalog

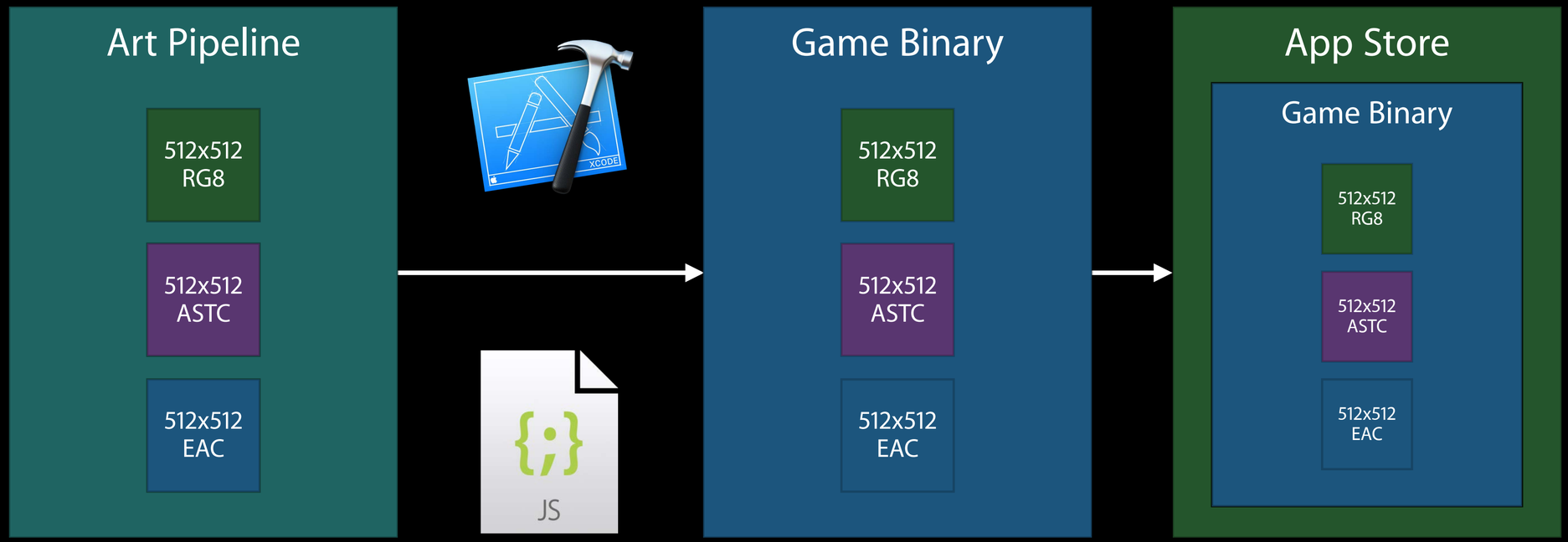
```
#import <UIKIt/NSDataAsset.h>

NSDataAsset *asset = [[NSDataAsset alloc] initWithName:@“NormalMaps”];
NSData *data = asset.data;
```

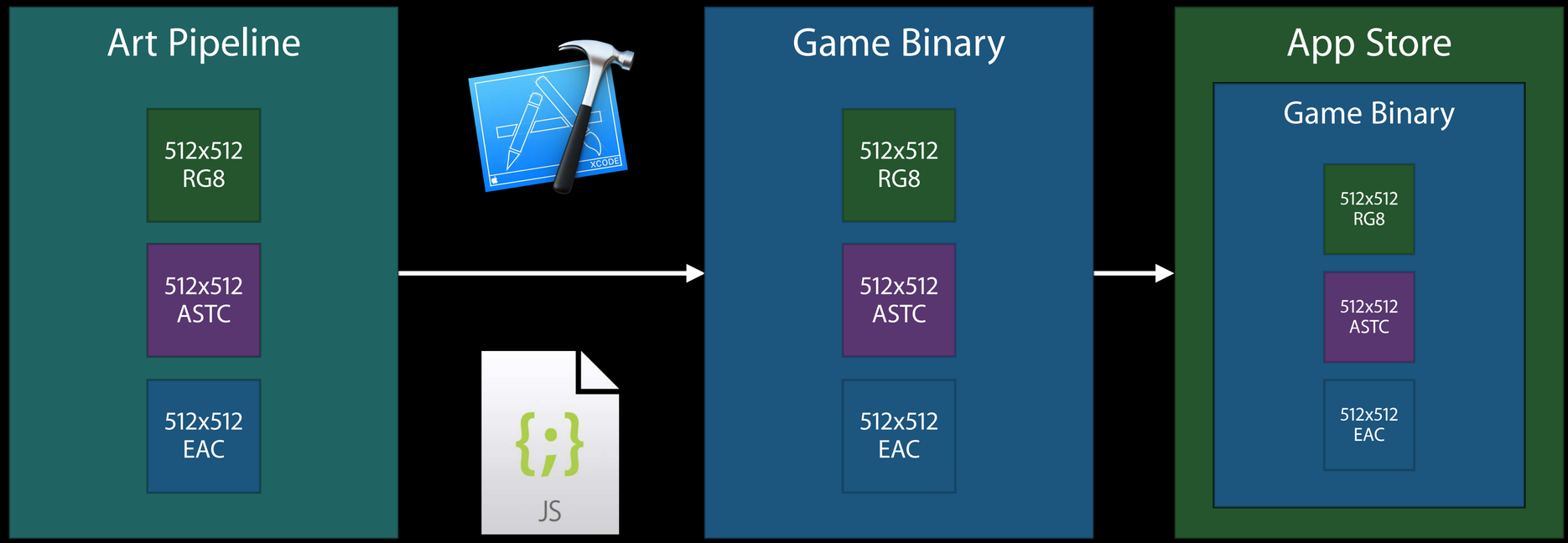

Art Pipeline
512x512
RG8
512x512
ASTC
512x512
EAC


Art Pipeline
512x512
RG8
512x512
ASTC
512x512
EAC
XCODE
JS
Game Binary
512x512
RG8
512x512
ASTC
512x512
EAC

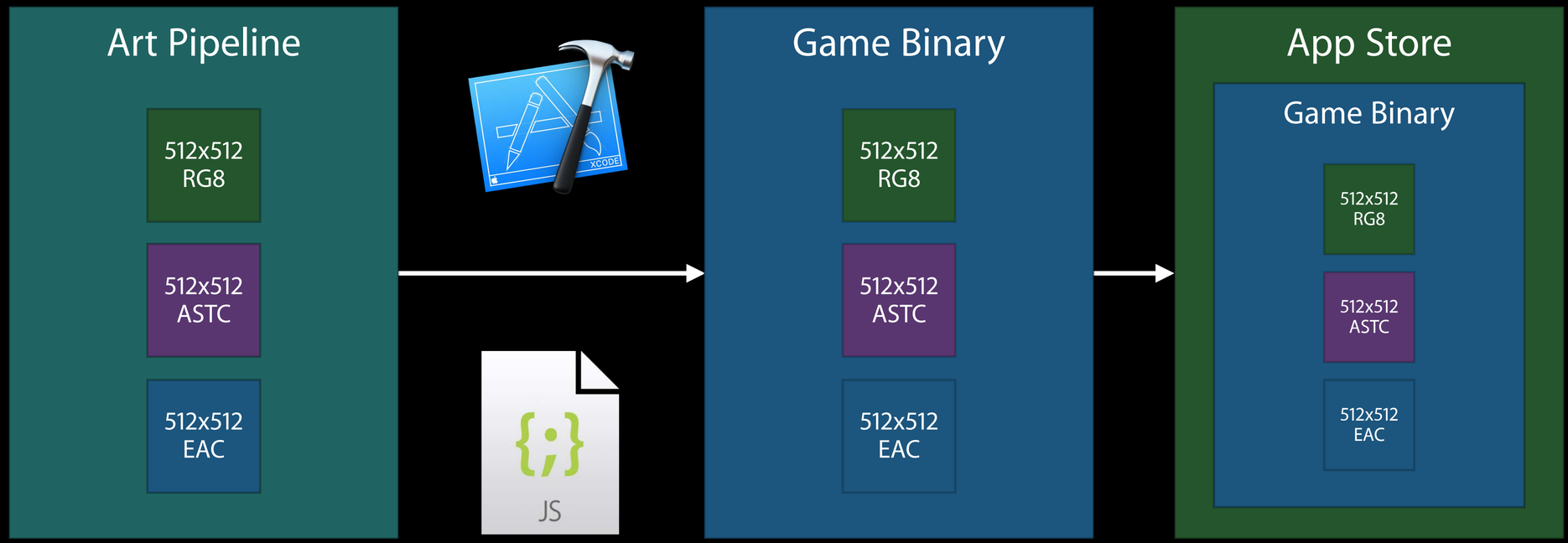

Art Pipeline
512x512
RG8
512x512
ASTC
512x512
EAC
XCODE
JS
Game Binary
512x512
RG8
512x512
ASTC
512x512
EAC
App Store

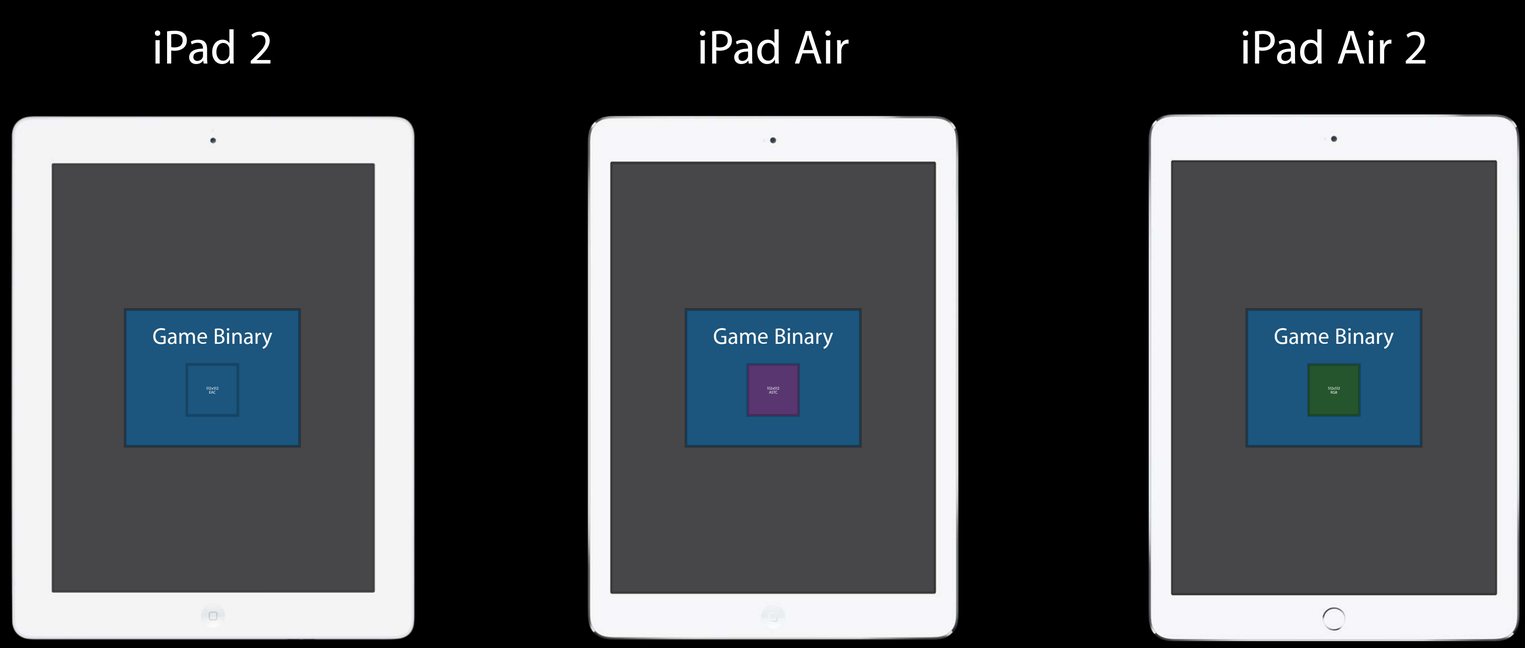

Art Pipeline
512x512
RG8
512x512
ASTC
512x512
EAC
XCODE
JS
Game Binary
512x512
RG8
512x512
ASTC
512x512
EAC
App Store
Game Binary
512x512
RG8
512x512
ASTC
512x512
EAC


Art Pipeline
512x512
RG8
512x512
ASTC
512x512
EAC
XCODE
JS
Game Binary
512x512
RG8
512x512
ASTC
512x512
EAC
App Store
Game Binary
512x512
RG8
512x512
ASTC
512x512
EAC


Art Pipeline
512x512
RG8
512x512
ASTC
512x512
EAC
XCODE
JS
Game Binary
512x512
RG8
512x512
ASTC
512x512
EAC
App Store
Game Binary
512x512
RG8
512x512
ASTC
512x512
EAC


iPad 2
iPad Air
iPad Air 2


Art Pipeline
512x512
RG8
512x512
ASTC
512x512
EAC
JS
Game Binary
512x512
RG8
512x512
ASTC
512x512
EAC
App Store
Game Binary
512x512
RG8
512x512
ASTC
512x512
EAC


iPad 2
Game Binary
iPad Air
Game Binary
iPad Air 2
Game Binary

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

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

WWDC15