

Metal 2 Optimization and Debugging

Session 607

Seth Sowerby, GPU Software

Metal 2

GPU-driven rendering

Platform feature alignment

Machine Learning acceleration

VR (macOS)

External GPU support (macOS)



Metal 2

GPU-driven rendering

Platform feature alignment

Machine Learning acceleration

VR (macOS)

External GPU support (macOS)

Advanced optimization tools



Metal 2 Developer Tools

Metal tools recap

Metal frame debugger enhancements

New GPU profiling tools



Metal Frame Debugger

Pulling your frame apart, bit by bit

Metal Frame Debugger

The story so far

The screenshot displays the Metal Frame Debugger interface for a captured GPU frame. The interface is divided into several main sections:

- Command Lists:** A tree view on the left shows the sequence of GPU commands, including state setters, texture loaders, and draw calls. The current frame is identified as 52 [drawInd...et:122532].
- Resource Details:** A central table lists resources used in the current command, categorized by type (Vertex, Fragment) and providing details like size, offset, and library.
- 3D View:** A wireframe rendering of the scene, showing a structure with columns and foliage. The wireframe is colored green. The view includes a depth map overlay and a color picker for the selected resource.
- Command Encoder Details:** A bottom panel provides detailed information about the selected command encoder, including its type, state, and various rendering parameters like viewport, scissor rect, blend color, and cull mode.

Label	Type	Size	Details
Vertex			
structure indices	Index	161 KB	Offset: 0x1dea4
structure vertices	Buffer 0	1.2 MB	Offset: 0x0
model matrices	Buffer 1	256 bytes	Offset: 0x0
gBufferVert	Vertex Function		Library 0x1740746...
Fragment			
foliage_normal	Texture 0	1024 x 1024	RGBA8Unorm
foliage_diffuse	Texture 1	1024 x 1024	RGBA8Unorm
foliage_specular	Texture 2	1024 x 1024	RGBA8Unorm
shadow map	Texture 3	1024 x 1024	Depth32Float
color 0	Color 0	1080 x 1920	BGRA8Unorm
color 1	Color 1	1080 x 1920	BGRA8Unorm
color 2	Color 2	1080 x 1920	R32Float
color 3	Color 3	1080 x 1920	BGRA8Unorm
depth	Depth	1080 x 1920	Depth32Float
stencil	Stencil	1080 x 1920	Stencil8
clear color buffer 2	Buffer 0	16 bytes	Offset: 0x0
gBufferFrag	Fragment Function		Library 0x1740746...

Metal Frame Debugger

The story so far

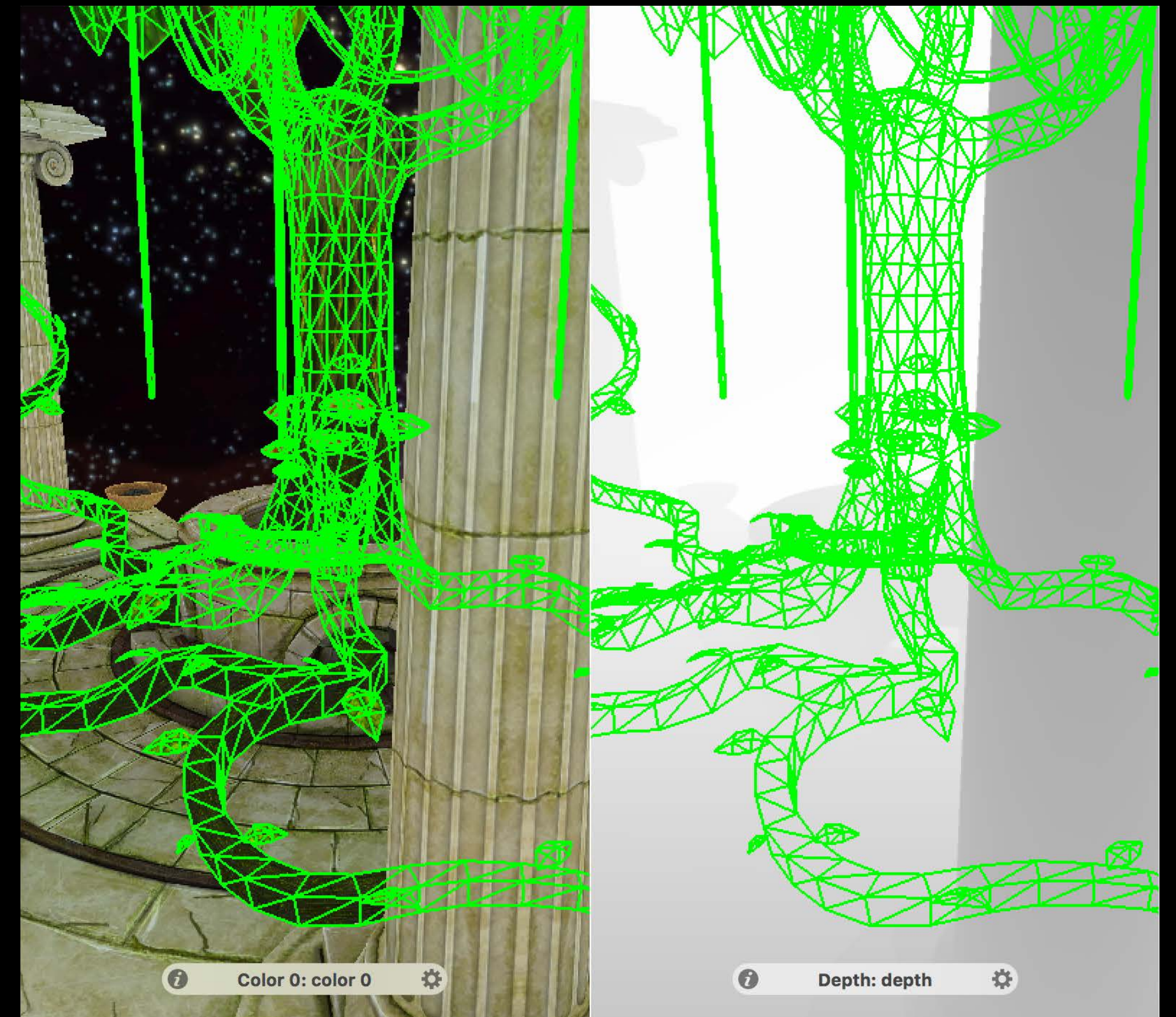
Fully featured frame debugger

Navigate through your workload

Inspect state and resource

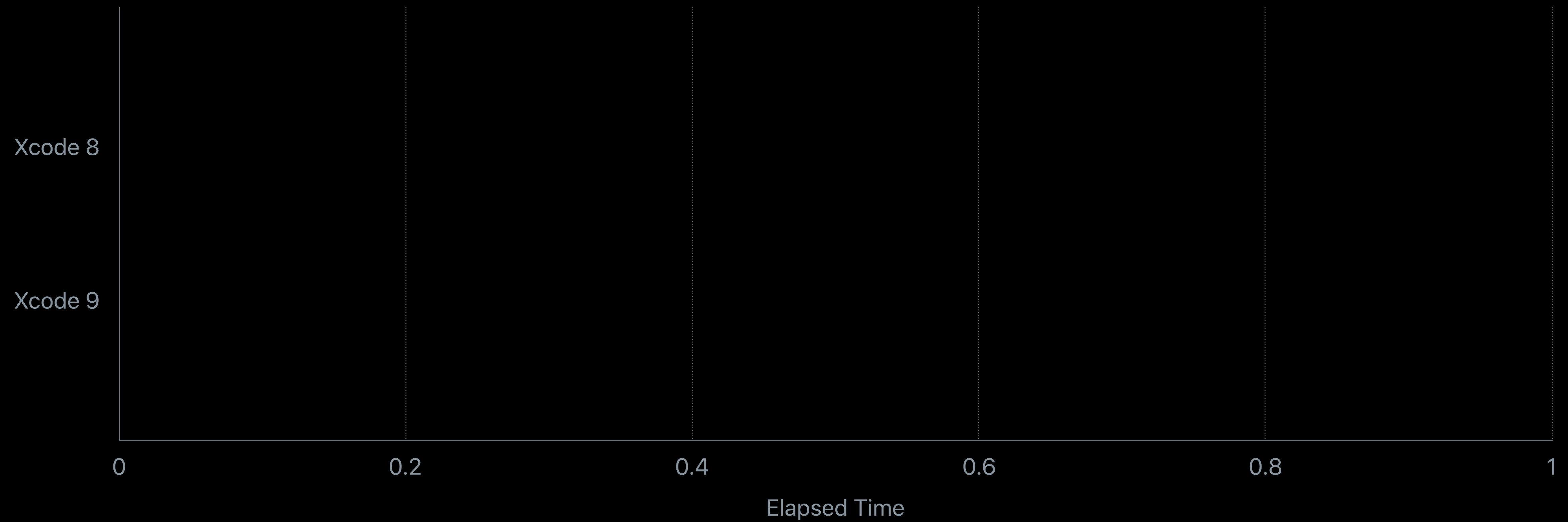
Debug graphics and compute

Integrated into Xcode



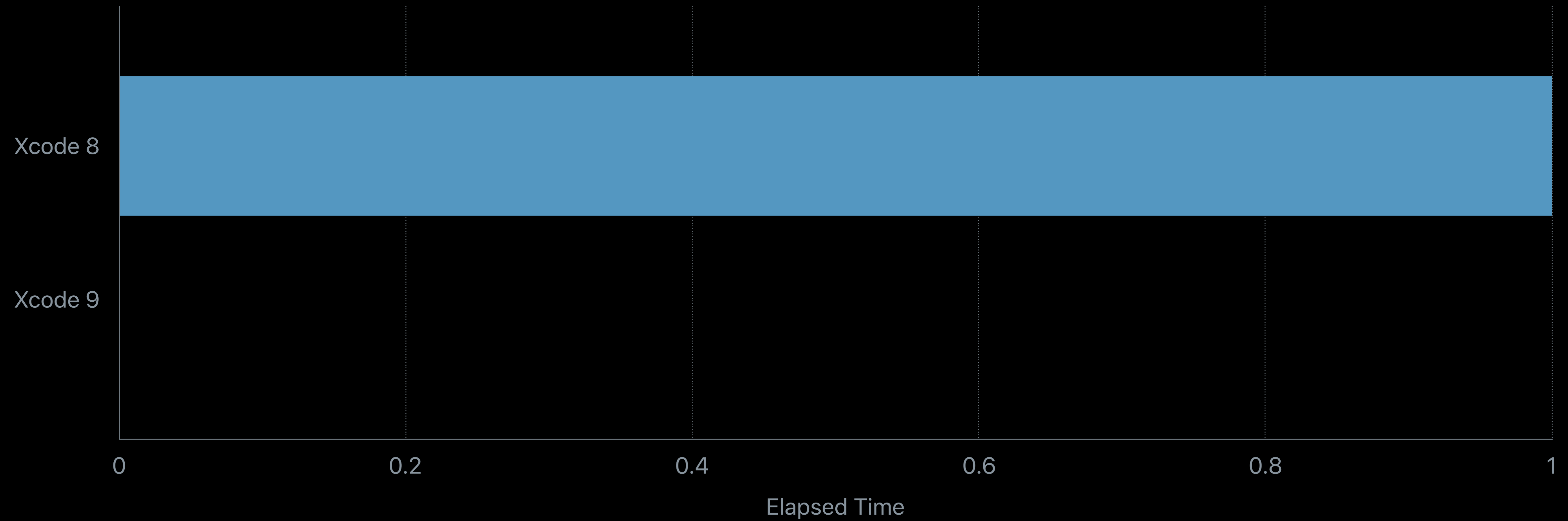
Improved Capture Performance

Up to 10x faster



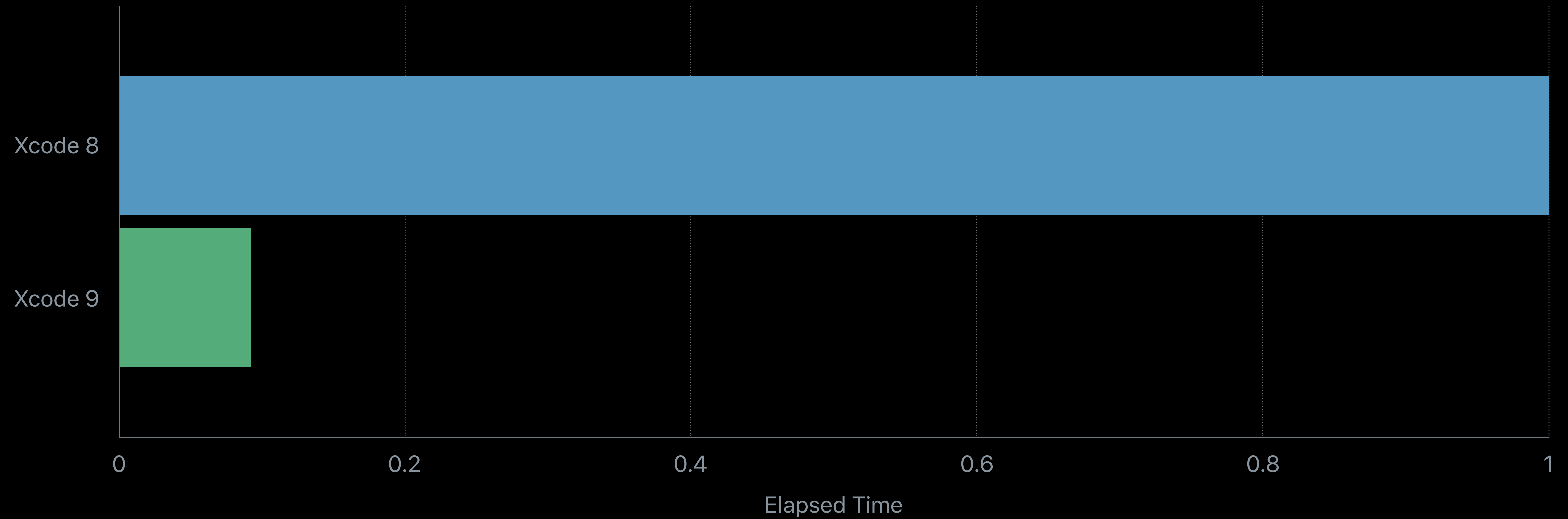
Improved Capture Performance

Up to 10x faster



Improved Capture Performance

Up to 10x faster



Full Metal 2 Support

NEW

Raster order groups

Sampler arrays

Viewport arrays

New pixel formats

New vertex array formats



Full Metal 2 Support

Argument buffers













































NEW

Row	Offset	Buffer uniforms	Texture colorTex	float4 ambientColor	Sampler colorSampler
0	0x0	Uniform Buffer 0x0	Stone	0.61 0.41 0.06 1.00	Repeat Sampler
1	0x30	Uniform Buffer 0x200	Water	0.61 0.41 0.06 1.00	Mirror Sampler
2	0x60	Uniform Buffer 0x400	Stone	0.14 0.03 0.83 1.00	Clamp Sampler
3	0x90	Uniform Buffer 0x600	Water	0.81 0.18 0.37 1.00	Repeat Sampler
4	0xC0	Uniform Buffer 0x800	Stone	0.27 0.74 0.36 1.00	Clamp Sampler
5	0xF0	Uniform Buffer 0xA00	Grass	0.81 0.18 0.37 1.00	Clamp Sampler
6	0x120	Uniform Buffer 0xC00	Water	0.14 0.03 0.83 1.00	Mirror Sampler
7	0x150	Uniform Buffer 0xE00	Grass	0.61 0.41 0.06 1.00	Repeat Sampler
8	0x180	Uniform Buffer 0x1000	Grass	0.27 0.74 0.36 1.00	Repeat Sampler
9	0x1B0	Uniform Buffer 0x1200	Grass	0.61 0.41 0.06 1.00	Mirror Sampler
10	0x1E0	Uniform Buffer 0x1400	Grass	0.81 0.18 0.37 1.00	Clamp Sampler
11	0x210	Uniform Buffer 0x1600	Grass	0.27 0.74 0.36 1.00	Repeat Sampler
12	0x240	Uniform Buffer 0x1800	Stone	0.61 0.41 0.06 1.00	Mirror Sampler
13	0x270	Uniform Buffer 0x1A00	Stone	0.81 0.18 0.37 1.00	Mirror Sampler
14	0x2A0	Uniform Buffer 0x1C00	Stone	0.14 0.03 0.83 1.00	Mirror Sampler
15	0x2D0	Uniform Buffer 0x1E00	Stone	0.81 0.18 0.37 1.00	Repeat Sampler
16	0x300	Uniform Buffer 0x2000	Water	0.14 0.03 0.83 1.00	Repeat Sampler
17	0x330	Uniform Buffer 0x2200	Water	0.61 0.41 0.06 1.00	Mirror Sampler
18	0x360	Uniform Buffer 0x2400	Water	0.81 0.18 0.37 1.00	Mirror Sampler
19	0x390	Uniform Buffer 0x2600	Stone	0.27 0.74 0.36 1.00	Mirror Sampler
20	0x3C0	Uniform Buffer 0x2800	Stone	0.81 0.18 0.37 1.00	Clamp Sampler
21	0x3F0	Uniform Buffer 0x2A00	Grass	0.61 0.41 0.06 1.00	Clamp Sampler
22	0x420	Uniform Buffer 0x2C00	Water	0.14 0.03 0.83 1.00	Mirror Sampler
23	0x450	Uniform Buffer 0x2E00	Water	0.81 0.18 0.37 1.00	Clamp Sampler
24	0x480	Uniform Buffer 0x3000	Grass	0.61 0.41 0.06 1.00	Mirror Sampler
25	0x4B0	Uniform Buffer 0x3200	Grass	0.27 0.74 0.36 1.00	Clamp Sampler
26	0x4E0	Uniform Buffer 0x3400	Stone	0.81 0.18 0.37 1.00	Mirror Sampler
27	0x510	Uniform Buffer 0x3600	Water	0.61 0.41 0.06 1.00	Mirror Sampler
28	0x540	Uniform Buffer 0x3800	Grass	0.14 0.03 0.83 1.00	Clamp Sampler
29	0x570	Uniform Buffer 0x3A00	Stone	0.14 0.03 0.83 1.00	Clamp Sampler

Full Metal 2 Support

Argument buffers

NEW

Buffer uniforms			Texture colorTex		float4 ambientColor							
Uniform Buffer	0x0	→	 Stone	→	0.61	0.41	0.06	1.00				
Uniform Buffer	0x200	→	 Water	→	0.61	0.41	0.06	1.00				
Uniform Buffer	0x400	→	 Stone	→	0.14	0.03	0.83	1.00				
Uniform Buffer	0x600	→	 Water	→	0.81	0.18	0.37	1.00				
Uniform Buffer	0x800	→	 Stone	→	0.27	0.74	0.36	1.00				
Uniform Buffer	0xA00	→	 Grass	→	0.81	0.18	0.37	1.00				
Uniform Buffer	0xC00	→	 Water	→	0.14	0.03	0.83	1.00				
Uniform Buffer	0xE00	→	 Grass	→	0.61	0.41	0.06	1.00				
Uniform Buffer	0x1000	→	 Grass	→	0.27	0.74	0.36	1.00				
Uniform Buffer	0x1200	→	 Grass	→	0.61	0.41	0.06	1.00				
Uniform Buffer	0x1400	→	 Grass	→	0.81	0.18	0.37	1.00				
Uniform Buffer	0x1600	→	 Grass	→	0.27	0.74	0.36	1.00				
Uniform Buffer	0x1800	→	 Stone	→	0.61	0.41	0.06	1.00				
Uniform Buffer	0x1A00	→	 Stone	→	0.81	0.18	0.37	1.00				
Uniform Buffer	0x1C00	→	 Stone	→	0.14	0.03	0.83	1.00				
23	0x450	Uniform Buffer	0x2E00	→	 Water	→	0.81	0.18	0.37	1.00	 Clamp Sampler	→
24	0x480	Uniform Buffer	0x3000	→	 Grass	→	0.61	0.41	0.06	1.00	 Mirror Sampler	→
25	0x4B0	Uniform Buffer	0x3200	→	 Grass	→	0.27	0.74	0.36	1.00	 Clamp Sampler	→
26	0x4E0	Uniform Buffer	0x3400	→	 Stone	→	0.81	0.18	0.37	1.00	 Mirror Sampler	→
27	0x510	Uniform Buffer	0x3600	→	 Water	→	0.61	0.41	0.06	1.00	 Mirror Sampler	→
28	0x540	Uniform Buffer	0x3800	→	 Grass	→	0.14	0.03	0.83	1.00	 Clamp Sampler	→
29	0x570	Uniform Buffer	0x3A00	→	 Stone	→	0.14	0.03	0.83	1.00	 Clamp Sampler	→

VR Support

NEW

Automatic support for SteamVR
View submitted surfaces in stereo



Improved Capture Workflow

Enhanced support for advanced workloads

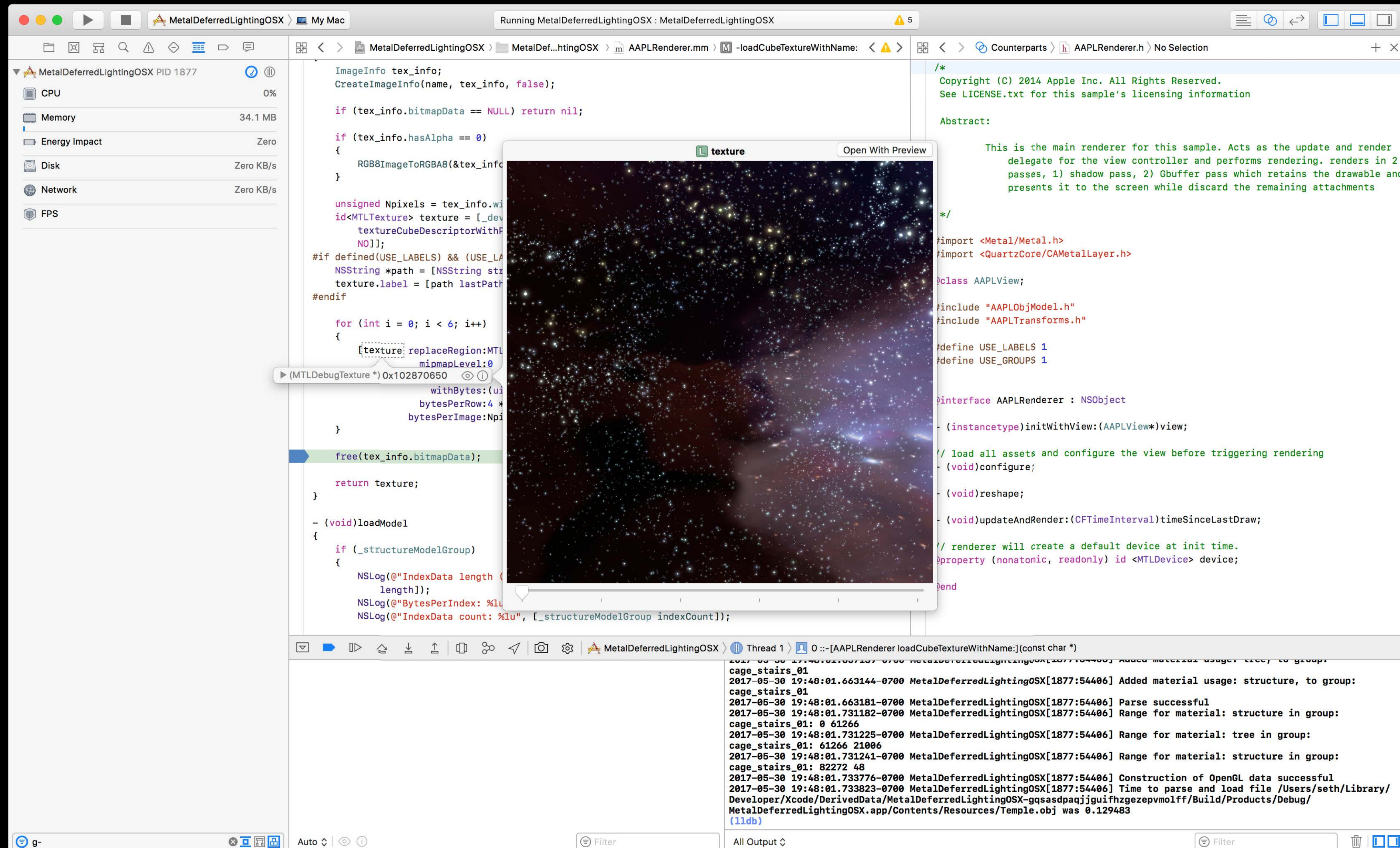


New lightweight capture API

- Capture the exact workload you're interested in
- Group workload into logical scopes
- Trigger capture programmatically

Metal Quick Looks

NEW



The screenshot displays the Xcode IDE interface for a project named "MetalDeferredLightingOSX". The interface is divided into several sections:

- Left Sidebar:** Shows system performance metrics for the running process (MetalDeferredLightingOSX PID 1877):
 - CPU: 0%
 - Memory: 34.1 MB
 - Energy Impact: Zero
 - Disk: Zero KB/s
 - Network: Zero KB/s
 - FPS: (not specified)
- Code Editor:** Displays the implementation of a texture loading function in `AAPLRenderer.mm`. The code includes comments and logic for creating an `MTLTexture` from a texture cube descriptor. A tooltip is visible over the `replaceRegion:MTLTexture` call, showing details like `midmapLevel:0` and `bytesPerImage:Npi`. Below the code, the `loadModel` method is partially visible, showing logging for index data.
- Preview Window:** A window titled "texture" displays a 3D scene of a starry space with a nebula, representing the rendered output of the application.
- Right Sidebar:** Shows the header file `AAPLRenderer.h` with a copyright notice and an abstract describing the renderer's role in performing rendering in two passes (shadow and Gbuffer).
- Bottom Panel:** The "All Output" window shows a log of system messages, including material usage reports and successful OpenGL data construction.

Metal Quick Looks

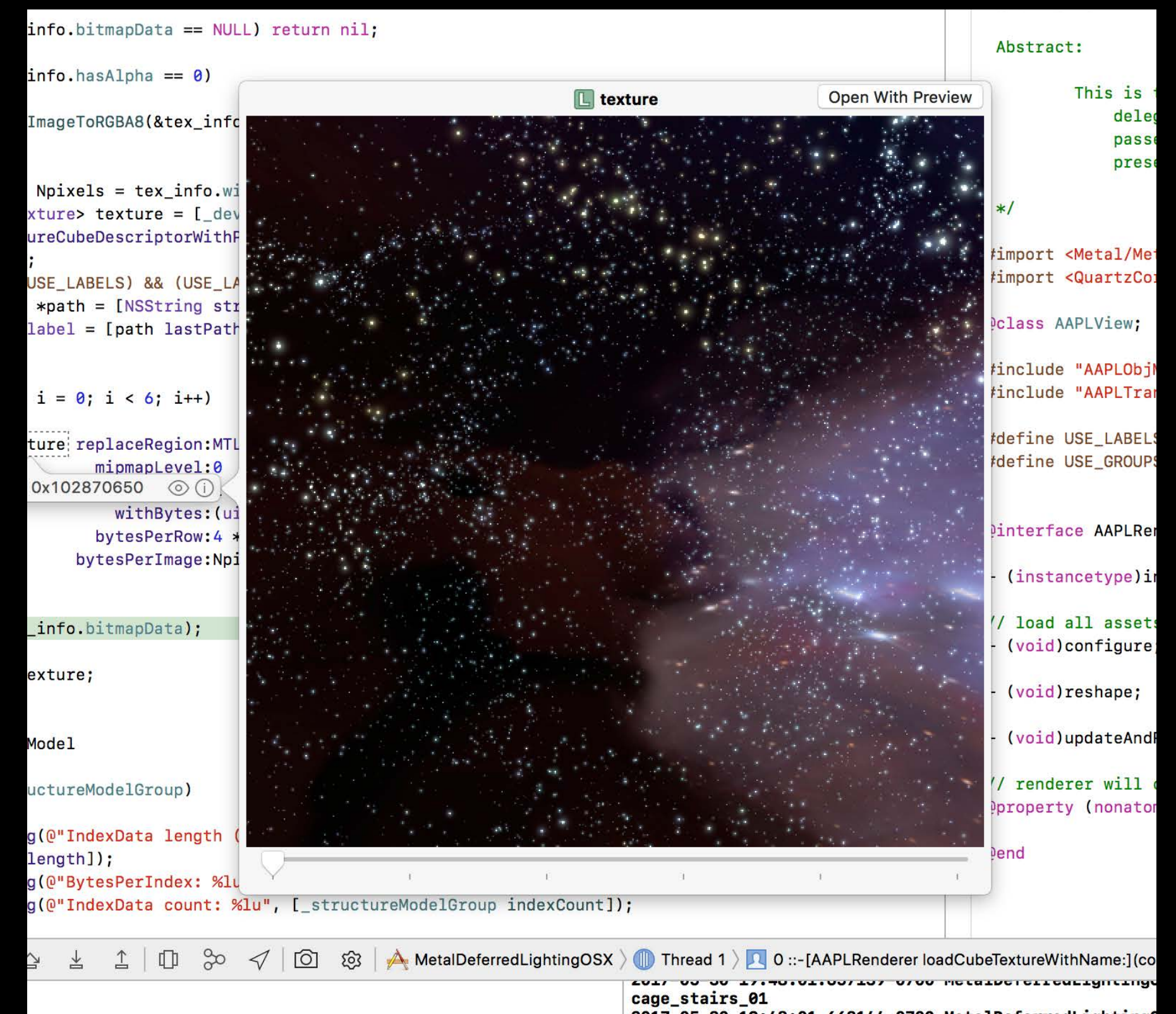
NEW

Lightweight Metal debugging in the CPU Debugger

- View textures, buffers and samplers inline

Use when the frame debugger is too invasive

- Resource loading and setup code



Advanced Filtering

NEW

Improved navigation and data mining for complex scenes

Autocomplete suggestions

Compound terms

The screenshot displays a GPU profiler interface for a scene titled "MetalDeferredLightingOSX Captured GPU Fr...". The interface is organized into several sections:

- FPS**: A section at the top, currently empty.
- GPU**: A section containing a hierarchical tree of GPU commands:
 - MyCmdBuf0** (expanded)
 - g-buffer** (expanded)
 - 20 g-buffer** <- [renderCommandEncoderWit...
 - g-buffer pass** (expanded)
 - skybox** (expanded)
 - 30 [drawPrimitives:TriangleSt...** 145.10 μs
 - 31 [drawPrimitives:TriangleStri...** 6.13 μs

Below the tree, there are three sections for autocomplete suggestions:

- Function**: skyboxVert, skyboxFrag
- Texture**: skybox.png
- Command Encoder**: skybox

At the bottom, there is a search bar with the text "g-buffer sk" and a filter dropdown set to "ALL". The search bar also includes a "COMMAND ENCODER" dropdown and a "sky" dropdown.

Pixel Inspection Finally!

NEW

The screenshot displays the GPU debugging interface for the game Obduction. The main window shows a 3D scene of a house with a circular pixel inspection tool overlaid on a portion of the house's roof. The tool displays a grid of color values: 87 194 238 255. The interface includes a command buffer log on the left, a resource table in the center, and a detailed view of the selected resource at the bottom.

Label	Type	Size	Details
▼ Vertex			
Buffer Ox108f18f90	Index	256 bytes	Offset: 0xc
Buffer Ox2f7f88990	Buffer 0	32 MB	Offset: 0x5b8e00
Buffer Ox2f7f88990	Buffer 30	32 MB	Offset: 0x5b8f00
Vertex Attributes	Vertex Attributes		
Main	Vertex Function		Library Ox108b720...
▼ Fragment			
Tonemap	Texture 0	1400 x 876	BGRA8Unorm
Texture Ox21ce37b10	Texture 1	1400 x 876	Depth24Unorm_Ste...
Texture Ox17b0c5860	Color 0	1400 x 875	BGRA8Unorm
Buffer Ox2f7f88990	Buffer 0	32 MB	Offset: 0x5b7f00
Buffer Ox2f7f88990	Buffer 1	32 MB	Offset: 0x5b8d00
Buffer Ox2f7f88990	Buffer 2	32 MB	Offset: 0x5b8c00
Buffer Ox2f7f88990	Buffer 3	32 MB	Offset: 0x5b8700
Sampler Ox10bceaa50	Sampler 0		
Sampler Ox10bceaa50	Sampler 1		
Main	Fragment Function		Library Ox11fb3a...

RenderPipelineState Ox23d2be550 Main - Main
RenderPipeline Performance 737.92 μs (0.8%)
Vertex Buffer 0 (MTLBuffer) Ox2f7f88990, Offset=0x005b8e00
Vertex Buffer 30 (MTLBuffer) Ox2f7f88990, Offset=0x005b8f00
Fragment Buffer 0 (MTLBuffer) Ox2f7f88990, Offset=0x005b7f00
Fragment Buffer 1 (MTLBuffer) Ox2f7f88990, Offset=0x005b8d00
Fragment Buffer 2 (MTLBuffer) Ox2f7f88990, Offset=0x005b8c00
Fragment Buffer 3 (MTLBuffer) Ox2f7f88990, Offset=0x005b8700
Fragment Texture 0 (MTLTexture) "Tonemap" (Ox3156f14c0) - 1400 x 876, BGRA8Unorm

Pixel Inspection

Finally!

NEW

Detailed inspection of individual pixels

Simultaneously inspect multiple attachments

Examine image elements in compute workloads



Inspect Vertex Attribute Outputs

NEW

Inspect output of your vertex shaders

Shown inline with vertex attribute inputs

in - float3 position			out - float4 position			
0.000	-2.000	0.000	-1.141	-1.618	6.493	6.586
-7e-08	-2.000	1.500	0.107	-3.282	6.539	6.633
-0.312	-2.000	1.467	-0.109	-3.483	6.759	6.852
0.000	-2.000	0.000	-1.141	-1.618	6.493	6.586
-0.312	-2.000	1.467	-0.109	-3.483	6.759	6.852
-0.610	-2.000	1.370	-0.371	-3.602	6.967	7.060
0.000	-2.000	0.000	-1.141	-1.618	6.493	6.586
-0.610	-2.000	1.370	-0.371	-3.602	6.967	7.060
-0.882	-2.000	1.214	-0.666	-3.635	7.154	7.247
0.000	-2.000	0.000	-1.141	-1.618	6.493	6.586
-0.882	-2.000	1.214	-0.666	-3.635	7.154	7.247

Inspect Vertex Attribute Outputs

NEW

in – float3 position			out – float4 position			
0.000	-2.000	0.000	-1.141	-1.618	6.493	6.586
-7e-08	-2.000	1.500	0.107	-3.282	6.539	6.633
-0.312	-2.000	1.467	-0.109	-3.483	6.759	6.852
0.000	-2.000	0.000	-1.141	-1.618	6.493	6.586
-0.312	-2.000	1.467	-0.109	-3.483	6.759	6.852
-0.610	-2.000	1.370	-0.371	-3.602	6.967	7.060
0.000	-2.000	0.000	-1.141	-1.618	6.493	6.586
-0.610	-2.000	1.370	-0.371	-3.602	6.967	7.060
-0.882	-2.000	1.214	-0.666	-3.635	7.154	7.247
0.000	-2.000	0.000	-1.141	-1.618	6.493	6.586
-0.882	-2.000	1.214	-0.666	-3.635	7.154	7.247

Demo

Max Christ, GPU Software

GPU Profiling

Finding the nano-second

Profile, Optimize, Repeat

Always nano-seconds to be found

Performance is crucial to games

Consistent fast frame rate

Get the most out of the GPU for the best looking game

Increase efficiency for longer gaming experience

Profile, Optimize, Repeat

Always nano-seconds to be found

Performance is crucial to games

Consistent fast frame rate

Get the most out of the GPU for the best looking game

Increase efficiency for longer gaming experience

Use the GPU profiling tools

Metal System Trace

Investigate timing issues

- CPU/GPU parallelism
- Frame rate stutters

Trace your Metal workloads through the system

- CPU to GPU to Display

Integrated into Instruments



Metal System Trace

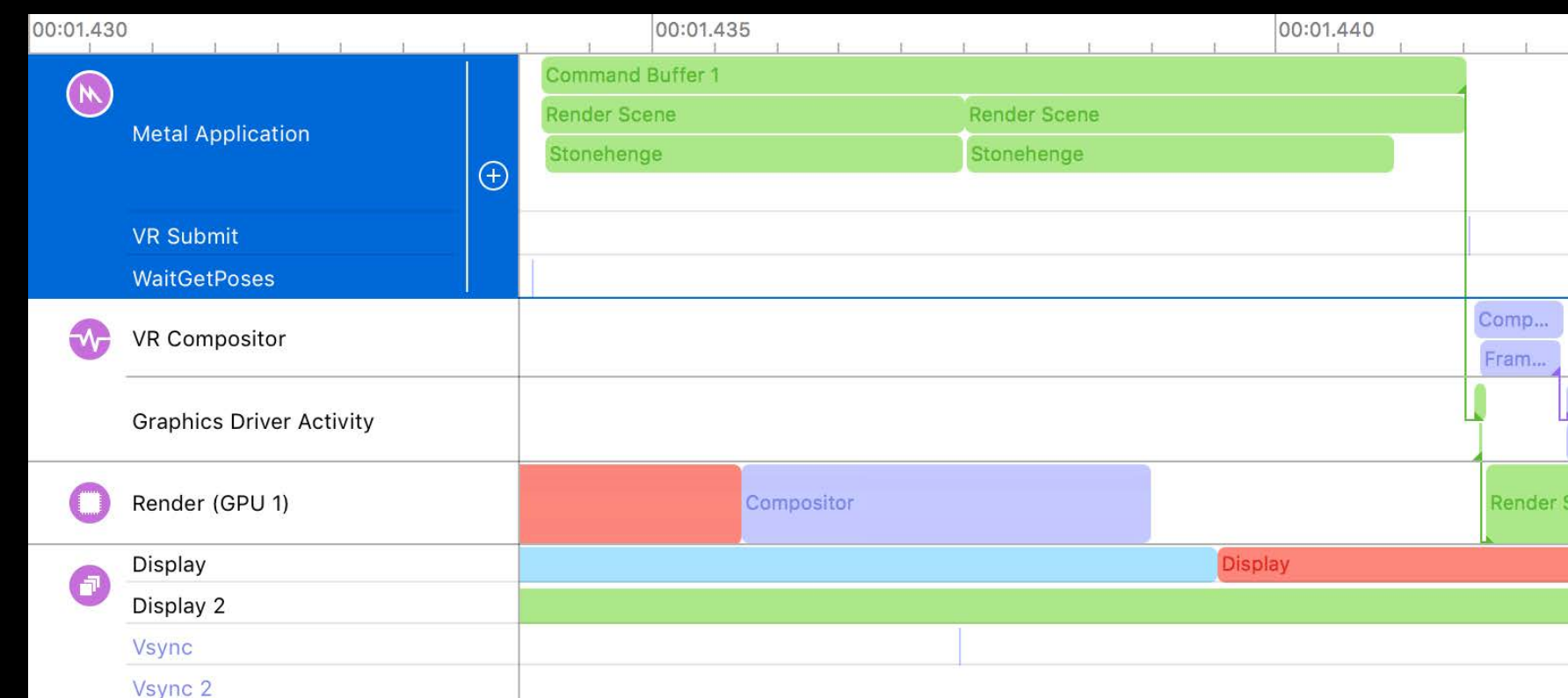
Support for VR apps

NEW

Support for VR specific tracepoints

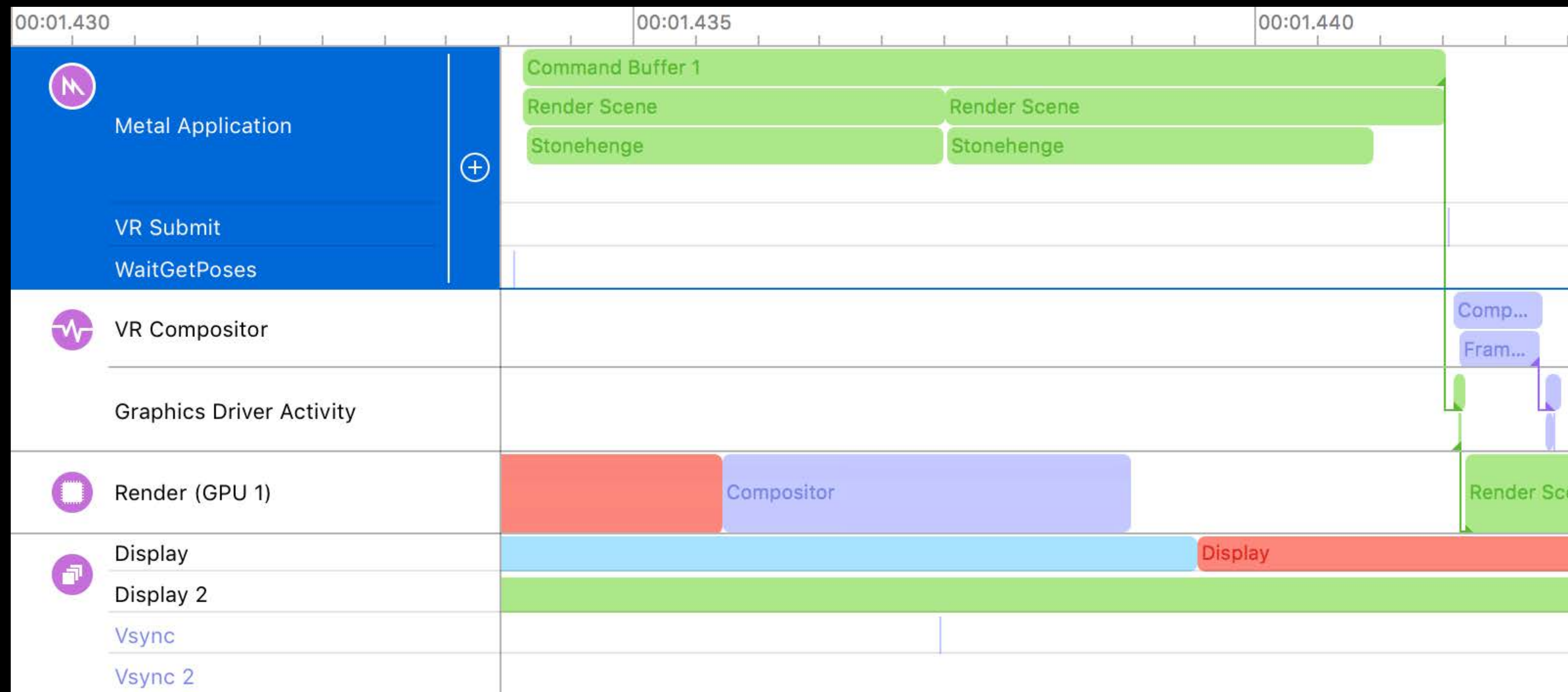
- App querying HMD data
- VR compositor behavior
- HMD surface

Trace from motion to photon



Metal System Trace

Support for VR apps



Metal System Trace

Other improvements

NEW

Support for ProMotion displays

Support for external GPUs

Improved Instruments integration

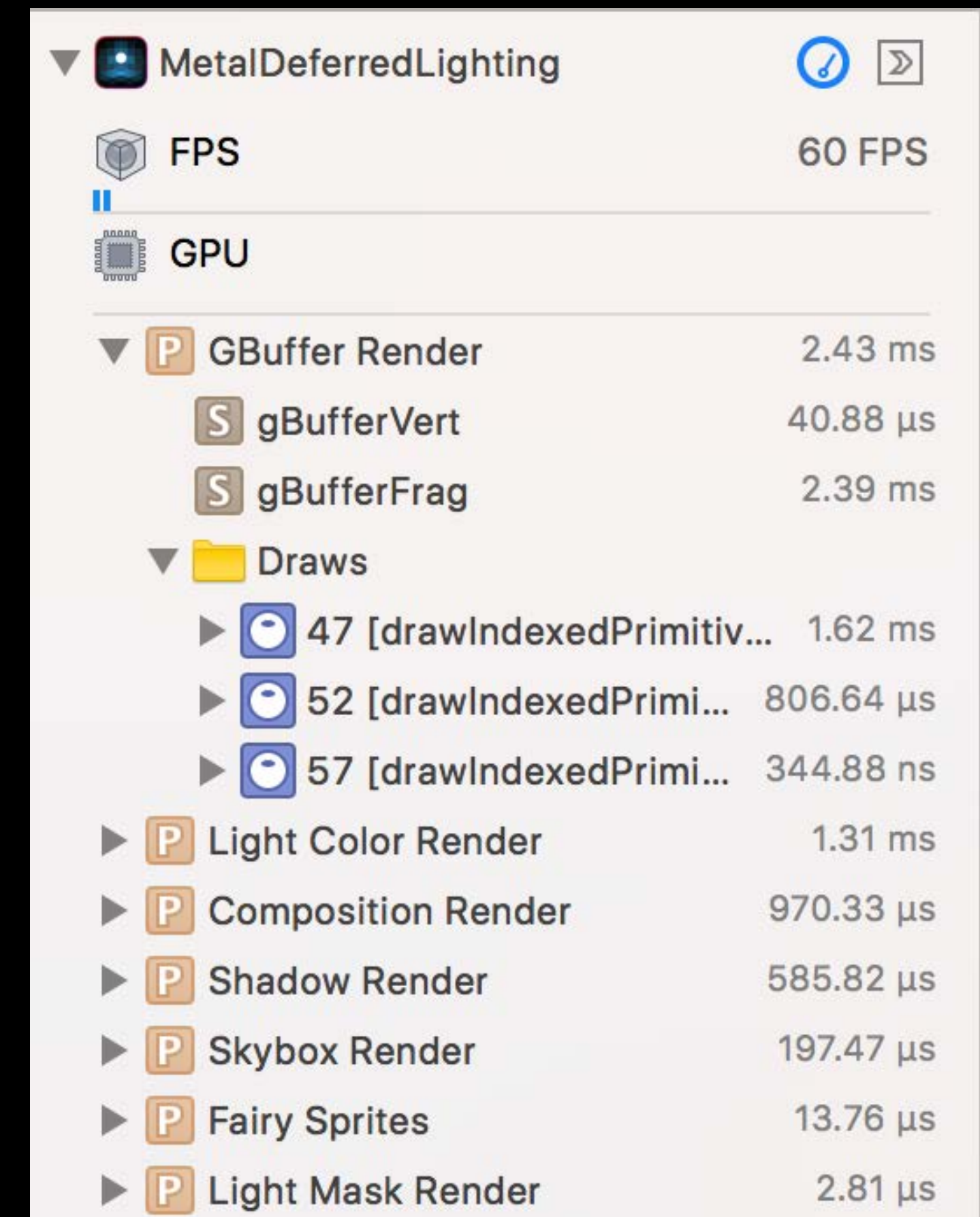


GPU Shader Profiler

The tool for probing shader performance

Integrated into Metal Frame Debugger

View shader time per draw/pipeline



The screenshot displays the GPU Shader Profiler interface for a MetalDeferredLighting scene. It shows a hierarchy of performance metrics, including FPS (60 FPS) and GPU time. The GPU section is expanded to show a list of rendering operations with their respective times. The 'GBuffer Render' category is expanded to show 'gBufferVert' (40.88 μs) and 'gBufferFrag' (2.39 ms). The 'Draws' category is also expanded to show three draw calls: '47 [drawIndexedPrimitiv...' (1.62 ms), '52 [drawIndexedPrimi...' (806.64 μs), and '57 [drawIndexedPrimi...' (344.88 ns). Other rendering operations include 'Light Color Render' (1.31 ms), 'Composition Render' (970.33 μs), 'Shadow Render' (585.82 μs), 'Skybox Render' (197.47 μs), 'Fairy Sprites' (13.76 μs), and 'Light Mask Render' (2.81 μs).

Category	Item	Time
MetalDeferredLighting	FPS	60 FPS
GPU	GBuffer Render	2.43 ms
	gBufferVert	40.88 μs
	gBufferFrag	2.39 ms
	Draws	
	47 [drawIndexedPrimitiv...	1.62 ms
	52 [drawIndexedPrimi...	806.64 μs
	57 [drawIndexedPrimi...	344.88 ns
	Light Color Render	1.31 ms
	Composition Render	970.33 μs
	Shadow Render	585.82 μs
	Skybox Render	197.47 μs
	Fairy Sprites	13.76 μs
	Light Mask Render	2.81 μs

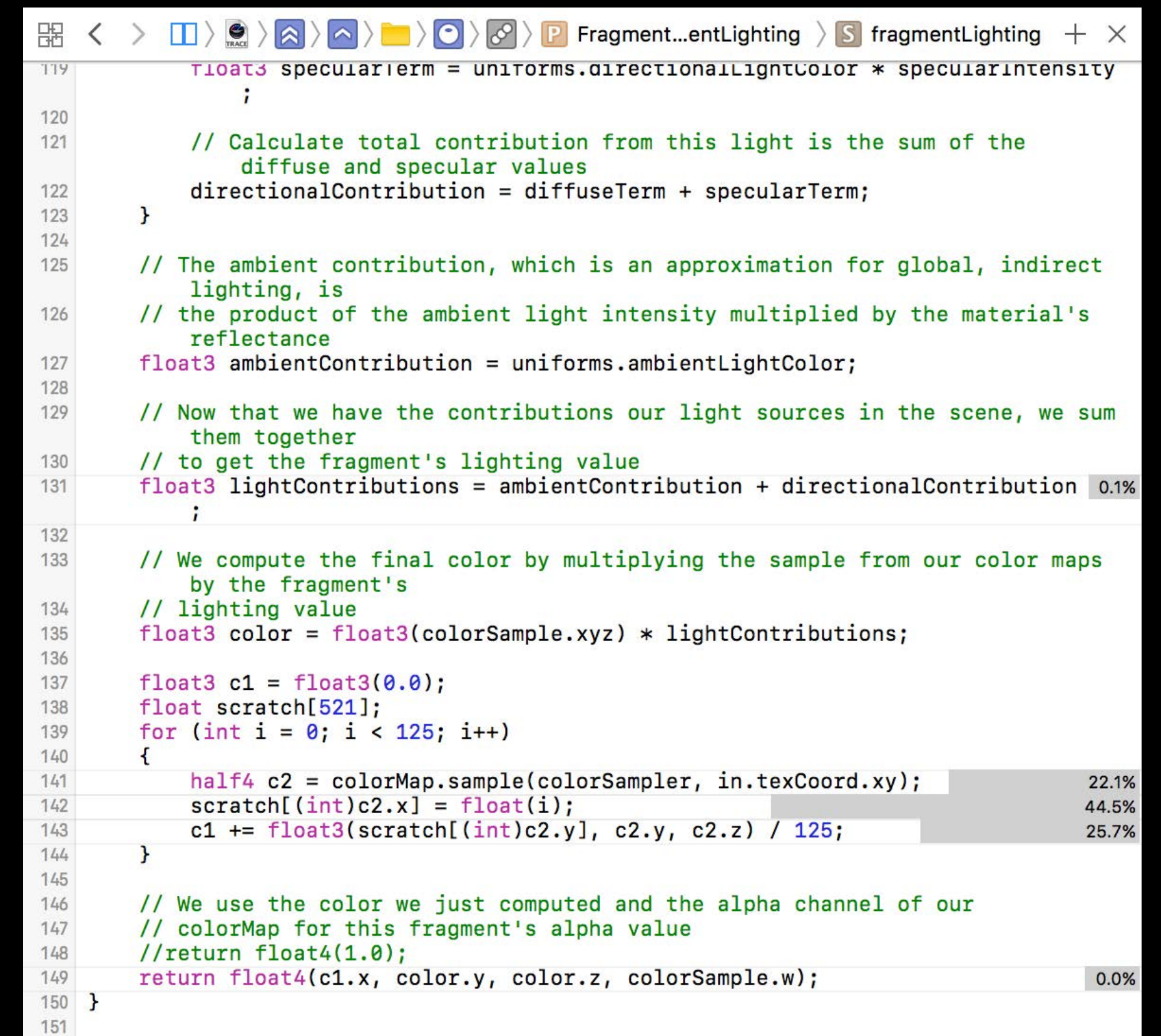
GPU Shader Profiler

The tool for probing shader performance

Integrated into Metal Frame Debugger

View shader time per draw/pipeline

Per-line execution cost (iOS/tvOS)



```
Fragment...entLighting > S fragmentLighting + X
119 float3 specularTerm = uniforms.directionalLightColor * specularIntensity
    ;
120
121 // Calculate total contribution from this light is the sum of the
    // diffuse and specular values
122 directionalContribution = diffuseTerm + specularTerm;
123 }
124
125 // The ambient contribution, which is an approximation for global, indirect
    // lighting, is
126 // the product of the ambient light intensity multiplied by the material's
    // reflectance
127 float3 ambientContribution = uniforms.ambientLightColor;
128
129 // Now that we have the contributions our light sources in the scene, we sum
    // them together
130 // to get the fragment's lighting value
131 float3 lightContributions = ambientContribution + directionalContribution 0.1%
    ;
132
133 // We compute the final color by multiplying the sample from our color maps
    // by the fragment's
134 // lighting value
135 float3 color = float3(colorSample.xyz) * lightContributions;
136
137 float3 c1 = float3(0.0);
138 float scratch[521];
139 for (int i = 0; i < 125; i++)
140 {
141     half4 c2 = colorMap.sample(colorSampler, in.texCoord.xy); 22.1%
142     scratch[(int)c2.x] = float(i); 44.5%
143     c1 += float3(scratch[(int)c2.y], c2.y, c2.z) / 125; 25.7%
144 }
145
146 // We use the color we just computed and the alpha channel of our
    // colorMap for this fragment's alpha value
147 //return float4(1.0);
148 //return float4(c1.x, color.y, color.z, colorSample.w); 0.0%
149 }
150 }
151 }
```


NEW

Metal Pipeline Statistics

Peeking inside the pipeline

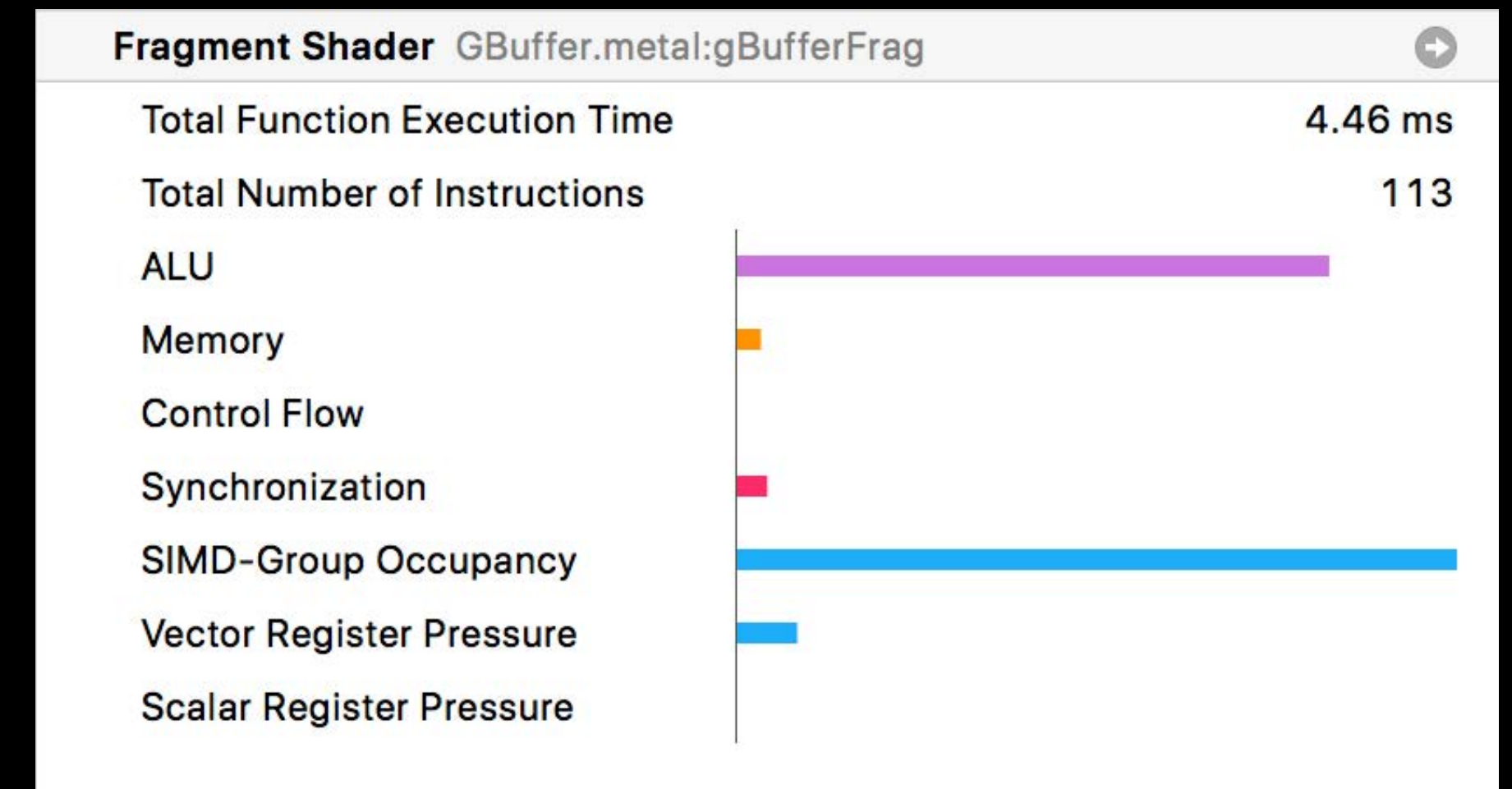
Metal Pipeline Statistics

Per-shader metrics

NEW

Compiler generated statistics

- Instruction count
- Instruction mix
- Register usage
- Occupancy



Metal Pipeline Statistics

Compiler remarks

NEW












Direct feedback from the GPU compiler

Actionable guidance on your shaders

Avoid performance hits

- Slow math usage
- Register spills
- Stack usage
- Other optimization opportunities

Remarks












- ▼  Register Spill 
Register allocation was exceeded and data was spilled into main GPU memory.
Reduce your function's register usage to avoid costly spills.
Spilling 1040 bytes - [Just-In-Time]:lightFrag 
- ▶  Dynamic Stack Store 
- ▶  Buffer Preloading Failed 
- ▶  Dynamic Stack Load 
- ▶  High Float-to-Half Ratio 

Metal Pipeline Statistics

Compiler remarks

NEW

Remarks

- ▼  Register Spill 
 - Register allocation was exceeded and data was spilled into main GPU memory.
Reduce your function's register usage to avoid costly spills.
 - Spilling 1040 bytes - [Just-In-Time]:lightFrag 
- ▶  Dynamic Stack Store 
- ▶  Buffer Preloading Failed 
- ▶  Dynamic Stack Load 
- ▶  High Float-to-Half Ratio 

Demo

Jose Enrique D'Arnaude del Castillo, GPU Software

GPU Counter Profiling

Showing you the nano-second

GPU Counter Profiling



NEW

GPU architecture is complex

- Pipeline of programmable and fixed function blocks
- Bottlenecks can occur anywhere
- Multiple simultaneous bottlenecks

NEW

“Minimize fixed-function bottlenecks while efficiently harnessing programmable blocks.”

Seth Sowerby, WWDC 2017

GPU Counter Profiling



NEW

New tool for deep GPU profiling

Detailed GPU performance counters

Common counter definitions across all GPUs

- No per-GPU learning curve

Integrated into GPU Frame Debugger

Metal2App My Mac Debugging GPU Frame 2 10

GPU Automatic Performance

Potential Hotspots/Bottlenecks

- Primitive culling very high - 100%
- Texture unit is very heavily stalled - 79%
- Texture cache miss rate is high - 74%

Check sampled textures have mipmaps

Counters	Draw 2094	Median	Max	Total
GPU Draw 2094				
GPU Time	1.59 ms	8.27 µs	2.36 ms	60.32 ms
Vertices				
Vertices	3	432	639,951	10,434,010
Vertices Reused	0%	65.13%	82.23%	-
Vertices Rendered	100%	35.04%	102.28%	-
Pixel per Vertex	409,500	5.42	409,500	-
Vertices per Second	2,466,565,000	10,114,320	7,557,846,000	-
Tessellator Busy	0%	0%	1%	-
Vertex Shader null:Main				
Vertex Shader Time	0.01%	26%	97.86%	-
VS Invocations	3	193	339,209	3,895,196
VS ALU Active Time	0.01%	0.21%	78.43%	-
VS Stall Time	0%	33.89%	95.98%	-
ALU Instructions/VS Invocation	17	61	724	252,789.1
Memory Instructions/VS Invocat...	3	16	55	39,155.22
Control Instructions/VS Invocati...	0	0	5	731.87
VS ALU to Memory Instructions...	5.67	5.52	15.95	-
Average VS SIMD-Groups	0.23	0.48	12.51	2,992.5
Average VS SIMD-Group Latency	421	1,069.5	8,365	2,441,580.25
VS SIMD-Group Memory Stall Ti...	1.43%	0%	87.62%	-
VS SIMD-Group Export Stall Time	3.81%	3.28%	59.67%	-
Parameter Cache Stall Time	0%	0%	63%	-
Primitives				
Primitives	1	144	213,316.99	3,478,583
Primitives Culled	100%	50%	119.24%	-
Primitives Culled (Zero-Area)	100%	0%	100%	-
Primitives Culled (Clipper)	0%	0%	100%	-
Primitives Clipped	100%	0%	100%	-
Primitives Rendered	300%	100%	300%	-
Pixels per Primitive	1,228,500	16.22	1,228,500	-
Hi-Z Test Fails	0%	2.74%	100%	-
PreZ Test Fails	0%	11.14%	99.97%	-
ROP Stall Time	26.83%	0%	80.63%	-
Primitives per Second	822,188,500	9,077,236	9,253,943,000	-
Fragment Shader null:Main				
Fragment Shader Time	98%	11.83%	99%	-
FS Invocations	1,225,000	41	1,319,390	64,202,240
FS ALU Active Time	29.82%	0.04%	93.4%	-
FS Stall Time	61.01%	17.79%	92.46%	-
ALU Instructions/FS Invocation	62	7	227	245,142.82

Filter

No Selection

GPU Counter Profiling

Graph view

NEW

Detailed GPU counter graphs

Top-level counters for each pipeline stage

More detail as you drill down



GPU Counter Profiling

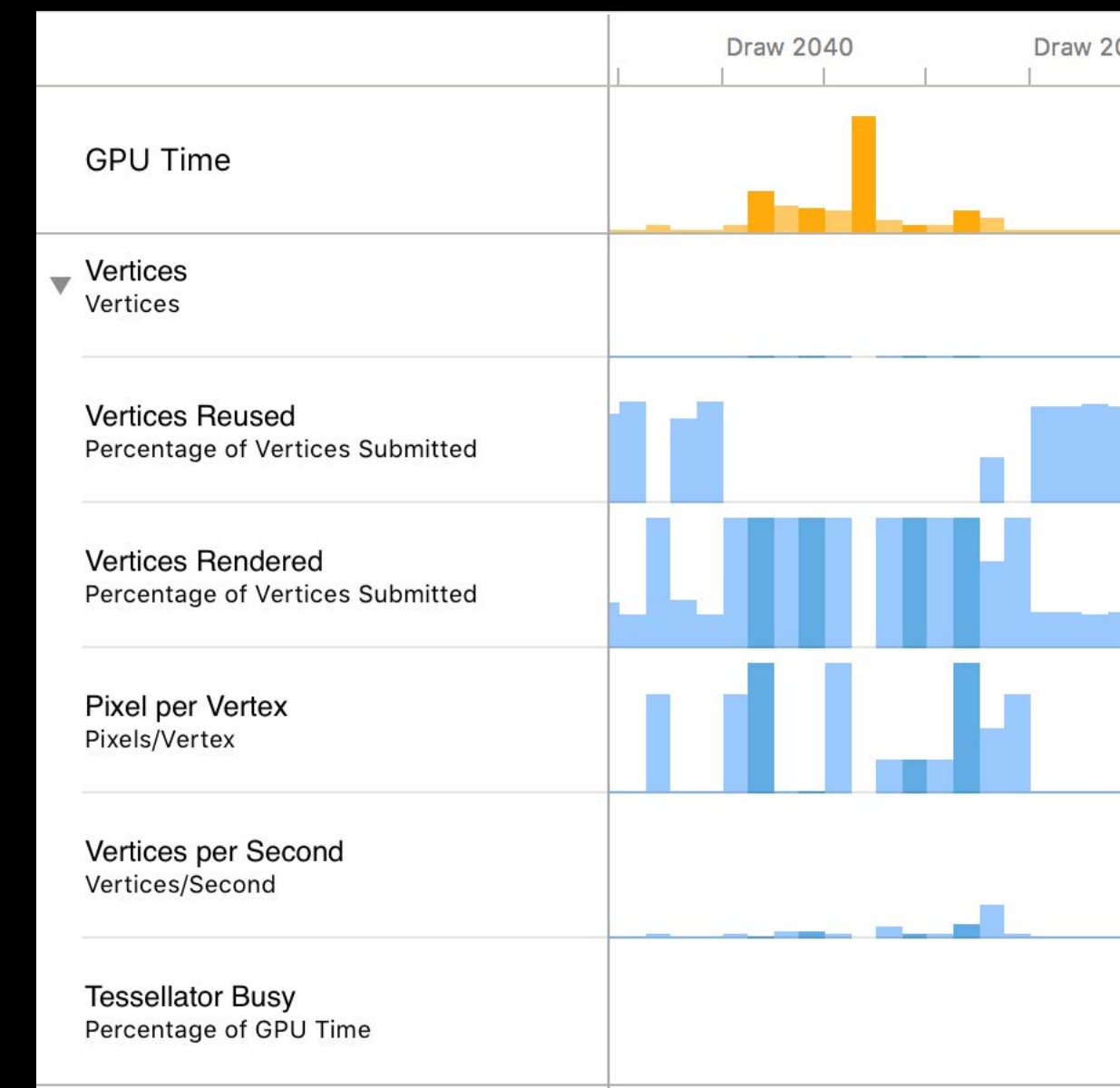
Graph view

NEW

Detailed GPU counter graphs

Top-level counters for each pipeline stage

More detail as you drill down



GPU Counter Profiling

Detail view

NEW

All counters from counter graph view

Presented in full detail

Frame median, max and total values

Fully filterable

Counters	Draw 2094	Median
GPU Draw 2094		
GPU Time	1.59 ms	8.27 μ s
Vertices		
Vertices	3	432
Vertices Reused	0%	65.13%
Vertices Rendered	100%	35.04%
Pixel per Vertex	409,500	5.42
Vertices per Second	2,466,565,000	10,114,320
Tessellator Busy	0%	0%
Vertex Shader null:Main		
Vertex Shader Time	0.01%	26%
VS Invocations	3	193
VS ALU Active Time	0.01%	0.21%
VS Stall Time	0%	33.89%
ALU Instructions/VS Invocation	17	61
Memory Instructions/VS Invocat...	3	16
Control Instructions/VS Invocati...	0	0
VS ALU to Memory Instructions...	5.67	5.52
Average VS SIMD-Groups	0.23	0.48
Average VS SIMD-Group Latency	421	1,069.5

GPU Counter Profiling

Bottleneck analysis

NEW



Advanced analysis of potential bottlenecks

Shared and GPU-specific analysis

Intuitive workflow to navigate direct

Detailed documentation

Potential Hotspots/Bottlenecks

- ▶  Primitive culling very high - 100%
-  Texture unit is very heavily stalled - 79%
- ▼  Texture cache miss rate is high - 74%
Check sampled textures have mipmaps 

Demo

Jose Enrique D'Arnaude del Castillo, GPU Software

GPU Counter Profiling



NEW

Yours to enjoy in Xcode 9 Beta 1

Available for all Metal capable GPUs

More recent GPUs have more counters available

Summary

Summary

Great enhancements to Metal Frame Debugger

Summary

Great enhancements to Metal Frame Debugger

Support for debugging and profiling VR apps

Summary

Great enhancements to Metal Frame Debugger

Support for debugging and profiling VR apps

Metal Pipeline Statistics

Summary

Great enhancements to Metal Frame Debugger

Support for debugging and profiling VR apps

Metal Pipeline Statistics

GPU Counter Profiling

More Information

<https://developer.apple.com/wwdc17/607>

Related Sessions

Introducing Metal 2

WWDC 2017

VR with Metal 2

WWDC 2017

Using Metal 2 for Compute

Grand Ballroom A

Thursday 4:10PM

Labs

Metal 2 Lab

Technology Lab F

Fri 9:00AM–12:00PM

