

#WWDC18

Metal Shader Debugging and Profiling

Session 608

Alp Yucebilgin, GPU Software

Metal

Metal API and language

MetalKit

Metal Performance Shaders

Metal developer tools



Metal

Metal API and language

MetalKit

Metal Performance Shaders

Metal developer tools

- Metal System Trace
- Metal Frame Debugger



Metal Frame Debugger

Step through API calls

Resource inspection

Shader edit and reload

GPU counters

Pipeline statistics

Integrated into Xcode

Metal Frame Debugger

NEW

Step through API calls

Resource inspection

Shader edit and reload

GPU counters

Pipeline statistics

Integrated into Xcode

Dependency viewer

Geometry viewer

Shader debugger

Enhanced shader profiler

Metal Frame Debugger

NEW

Step through API calls

Resource inspection

Shader edit and reload

GPU counters

Pipeline statistics

Integrated into Xcode

Dependency viewer

Geometry viewer

Shader debugger

Enhanced shader profiler

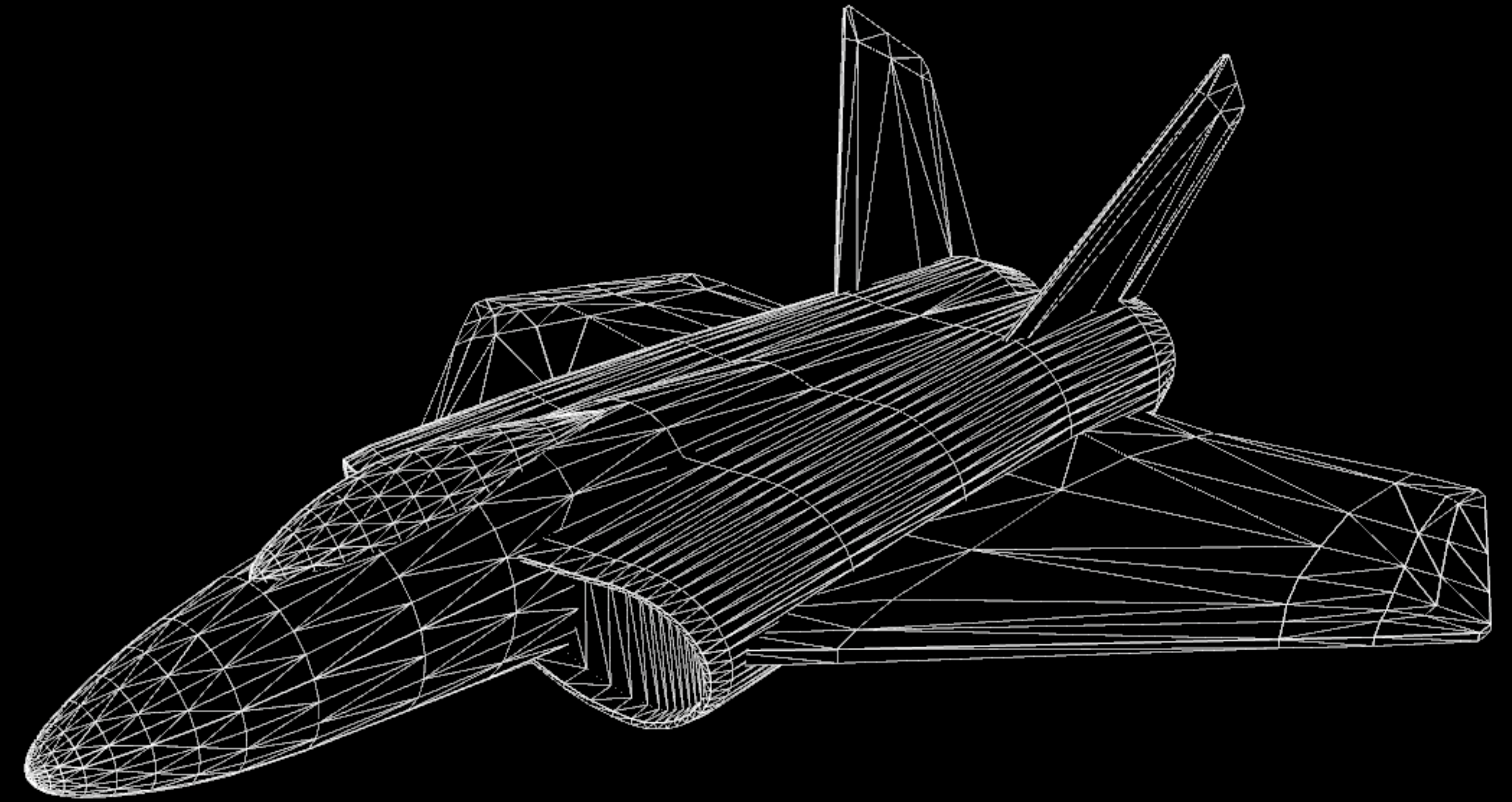
Investigating Vertex Issues with Geometry Viewer

Things to Watch in Vertex Stage

Vertex inputs

Indices

Output




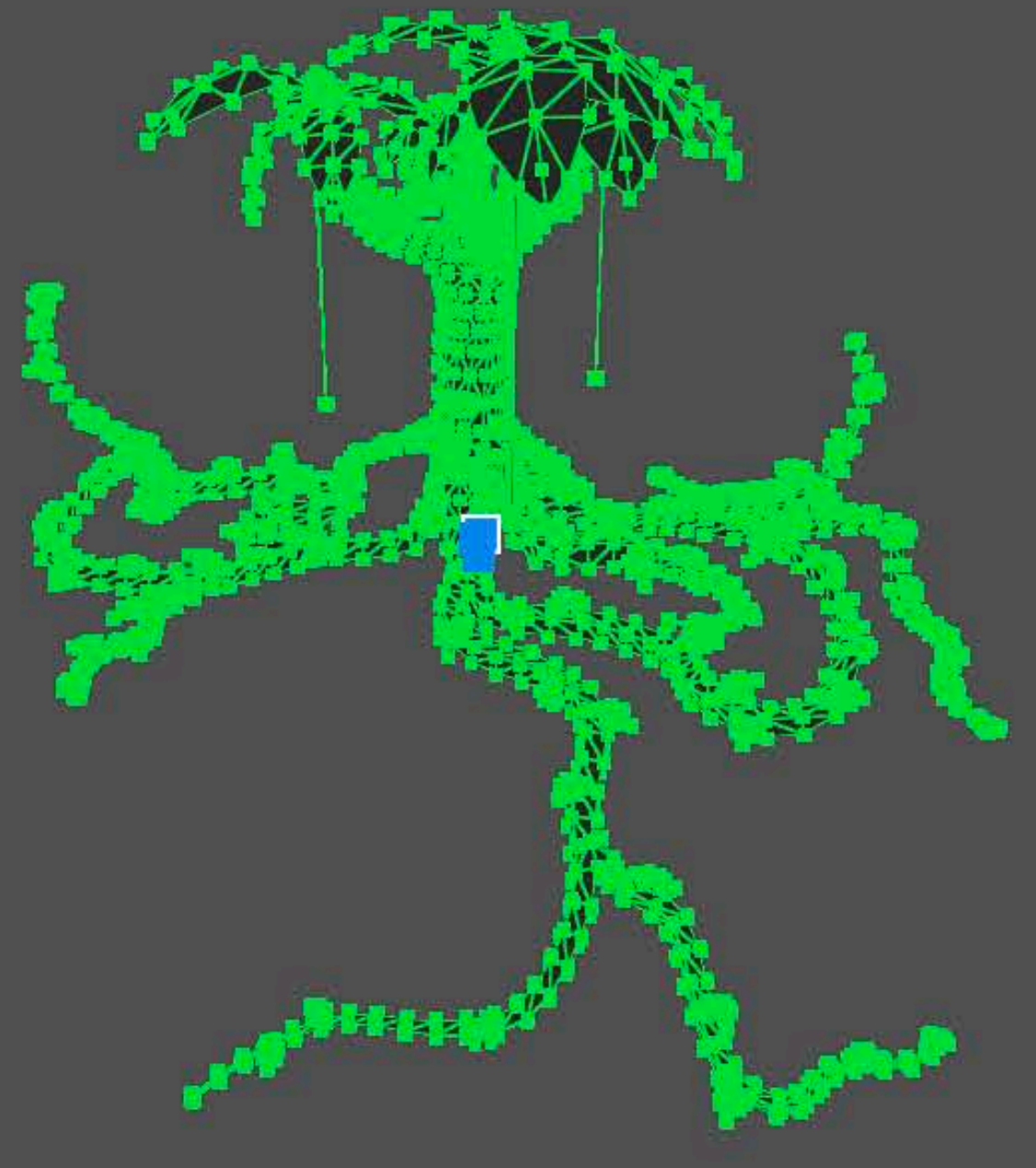
Deferred Lighting - Debugging GPU Frame

46 [drawIndexedPrimitives:Triangle i...J-Indices indexBufferOffset:245064]

Deferred Lighting Capture... 55 FPS

GPU

- 0 0x600000d59ec0 <- [MTLLaye...
- 1 0x100f2c460 <- [MTLDrawable...
- MyCommand
 - 2 MyCommand <- [0x100a0acf...
 - 3 [setLabel:"MyCommand"]
 - 4 [addCompletedHandler:0x7ff...
 - Shadow Map Pass
 - RenderCommandEncoder 0x10...
 - 28 0x1038bbe00 <- [render...
 - Draw G-Buffer
 - 29 [pushDebugGroup:"Dr...
 - 30 [setCullMode:Back]
 - 31 [setRenderPipelineStat...
 - 32 [setDepthStencilState:...
 - 33 [setStencilReferenceV...
 - 34 [setVertexBuffer:Unifo...
 - 35 [setFragmentBuffer:Un...
 - 36 [setFragmentTexture:S...
 - 37 [setVertexBuffer:0x10...
 - 38 [setVertexBuffer:0x10...
 - 39 [setFragmentTexture:T...
 - 40 [setFragmentTexture:T...
 - 41 [setFragmentTexture:T...
 - 42 [drawIndexedPrimitive...
 - 43 [setFragmentTexture:T...
 - 44 [setFragmentTexture:T...
 - 45 [setFragmentTexture:T...
 - 46 [drawIndexedPrimitive...
 - 47 [setFragmentTexture:T...
 - 48 [setFragmentTexture:T...
 - 49 [setFragmentTexture:T...
 - 50 [drawIndexedPrimitive...
 - 51 [popDebugGroup]
 - 52 [endEncoding]
 - RenderCommandEncoder 0x10...
 - 118 [presentDrawable:0x6000...
 - 119 [commit]

Color 1: Albedo + Shadow GBuffer

Index	in - uint Vertex	in - float3 position		out - float4 position				
0	15163	-4.86840	187.189	44.0387	0.518	-4.114	117.3	117.4
1	15164	0.24840	178.664	35.6922	-0.026	-4.784	118.4	118.5
2	15165	-3.60920	167.483	42.7444	0.384	-6.841	118.2	118.3
3	15165	-3.60920	167.483	42.7444	0.384	-6.841	118.2	118.3
4	15166	-6.85420	174.558	50.0506	0.729	-6.309	117.2	117.3

Perspective zNear 0.1 zFar 100.0 FOV 65.0 Debug

- "G-buffer Creation" (0x100a06a50) gbuffer_vertex - gbuffer_fragment
- RenderPipeline Performance Unavailable
- Vertex Buffer 0 (MTLBuffer) 0x100905250
- Vertex Buffer 1 (MTLBuffer) 0x1009053a0
- Vertex Buffer 2 (MTLBuffer) "UniformBuffer1" (0x100e06610)
- Fragment Buffer 2 (MTLBuffer) "UniformBuffer1" (0x100e06610)
- Fragment Texture 0 (MTLTexture) "Temple/FoliageBaseColorMap" (0x100a039e0) - 1024 x 1024, RGBA8Unorm_sRGB
- Fragment Texture 1 (MTLTexture) "Temple/FoliageSpecularMap" (0x100e06db0) - 1024 x 1024, RGBA8Unorm
- Fragment Texture 2 (MTLTexture) "Temple/FoliageNormalMap" (0x100f1a000) - 1024 x 1024, RGBA8Unorm
- Fragment Texture 3 (MTLTexture) "Shadow Map" (0x100f18350) - 2048 x 2048, Depth32Float
- Function = (MTLFunction) "gbuffer_fragment"


Geometry Viewer

NEW

Visualize the post-vertex transform data in 3D

Access to all vertex data

Per-draw call view

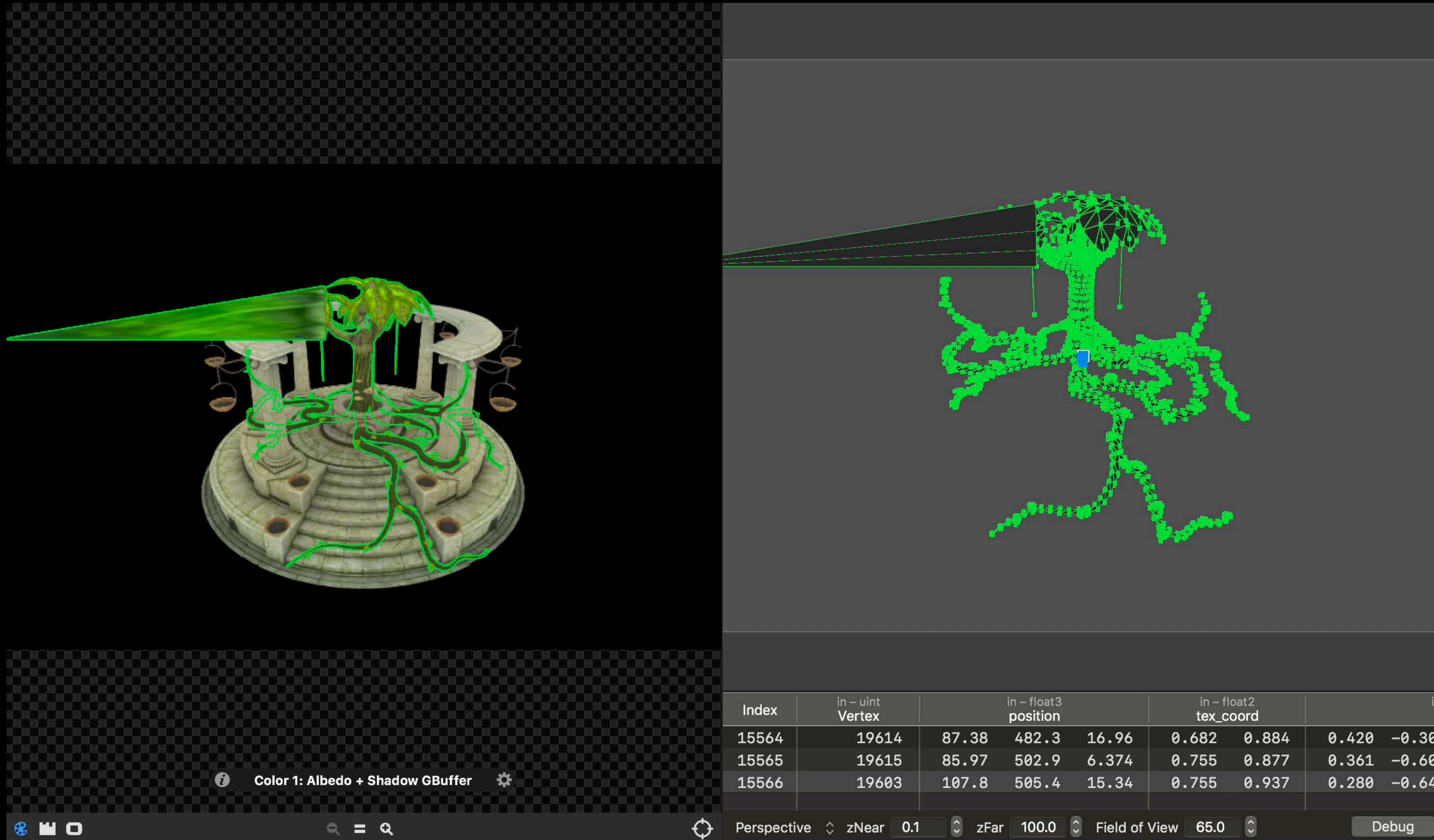


A 3D visualization of a tree model rendered in a wireframe style. The tree is composed of numerous vertices and edges, colored in a vibrant cyan. A single vertex on the trunk of the tree is highlighted with a small blue square, indicating it is the selected vertex for the data table below.

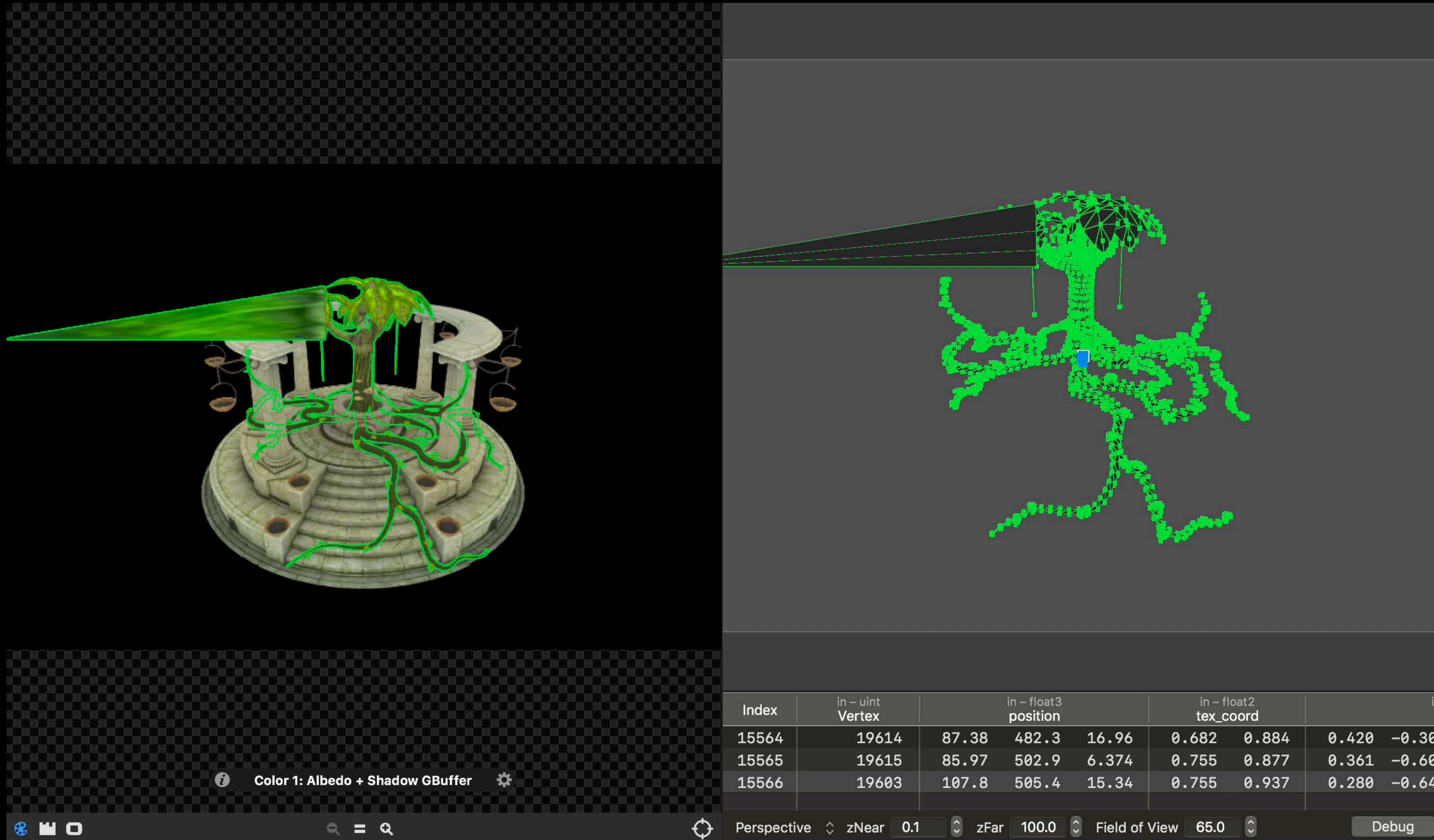
Index	in - uint Vertex	in - float3 position			out - float4 position			
0	15163	-4.86840	187.189	44.0387	0.518	-4.114	117.3	117.4
1	15164	0.24840	178.664	35.6922	-0.026	-4.784	118.4	118.5
2	15165	-3.60920	167.483	42.7444	0.384	-6.841	118.2	118.3
3	15165	-3.60920	167.483	42.7444	0.384	-6.841	118.2	118.3
4	15166	-6.85420	174.558	50.0506	0.729	-6.309	117.2	117.3

Perspective zNear 0.1 zFar 100.0 FOV 65.0

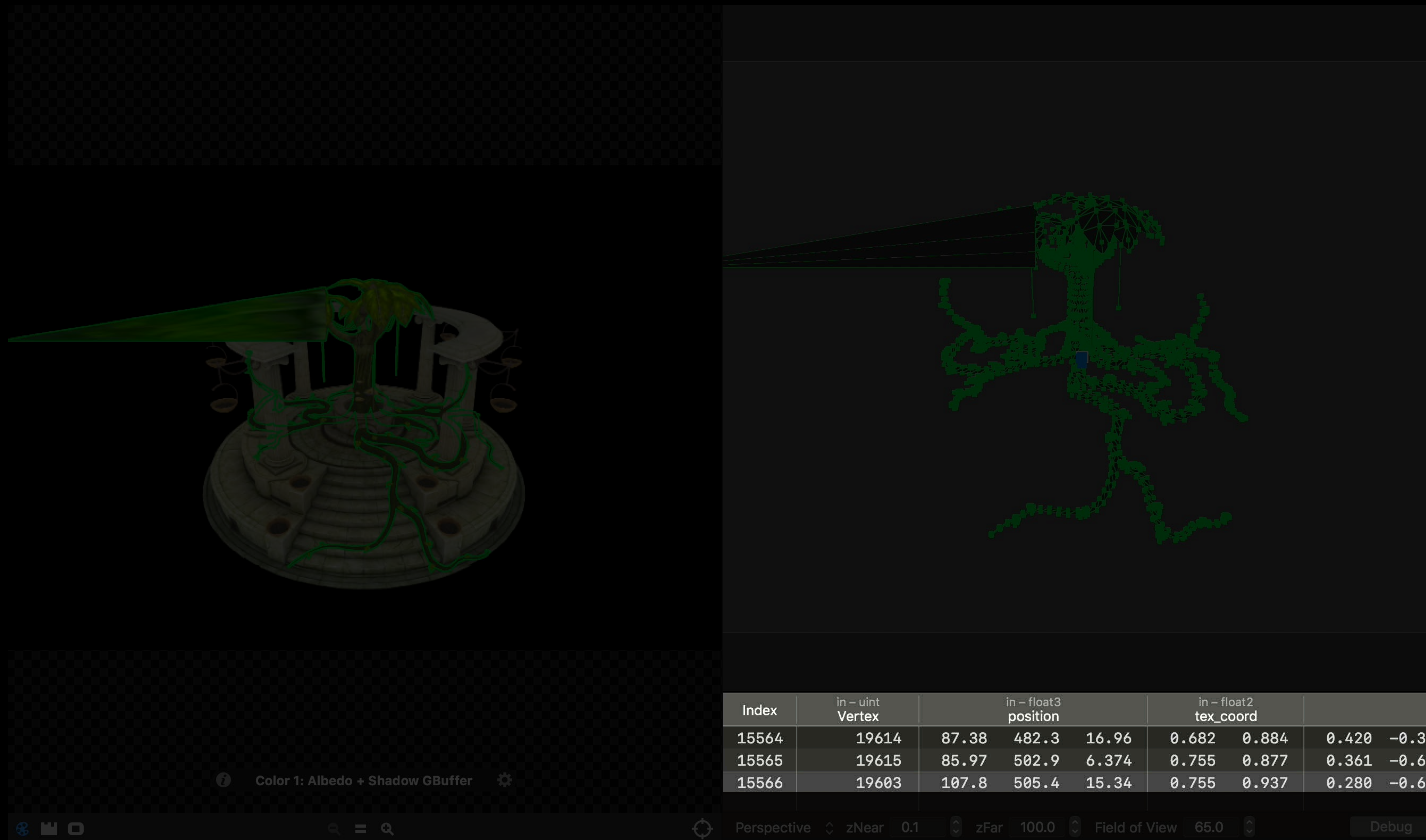
Visibly Wrong Triangles



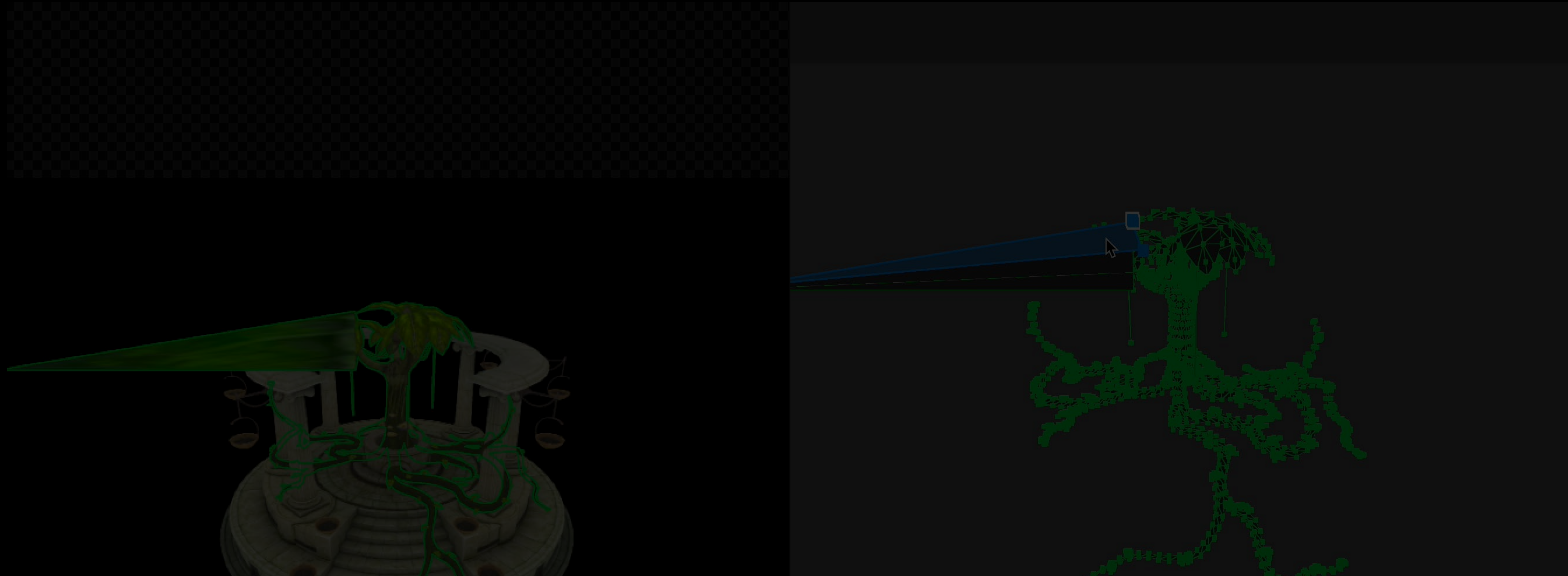
Visibly Wrong Triangles



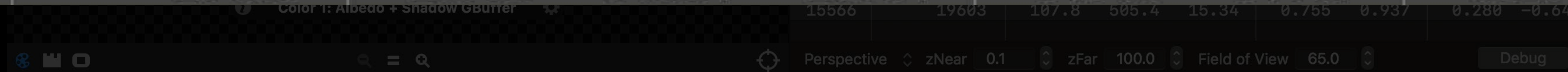
Visibly Wrong Triangles



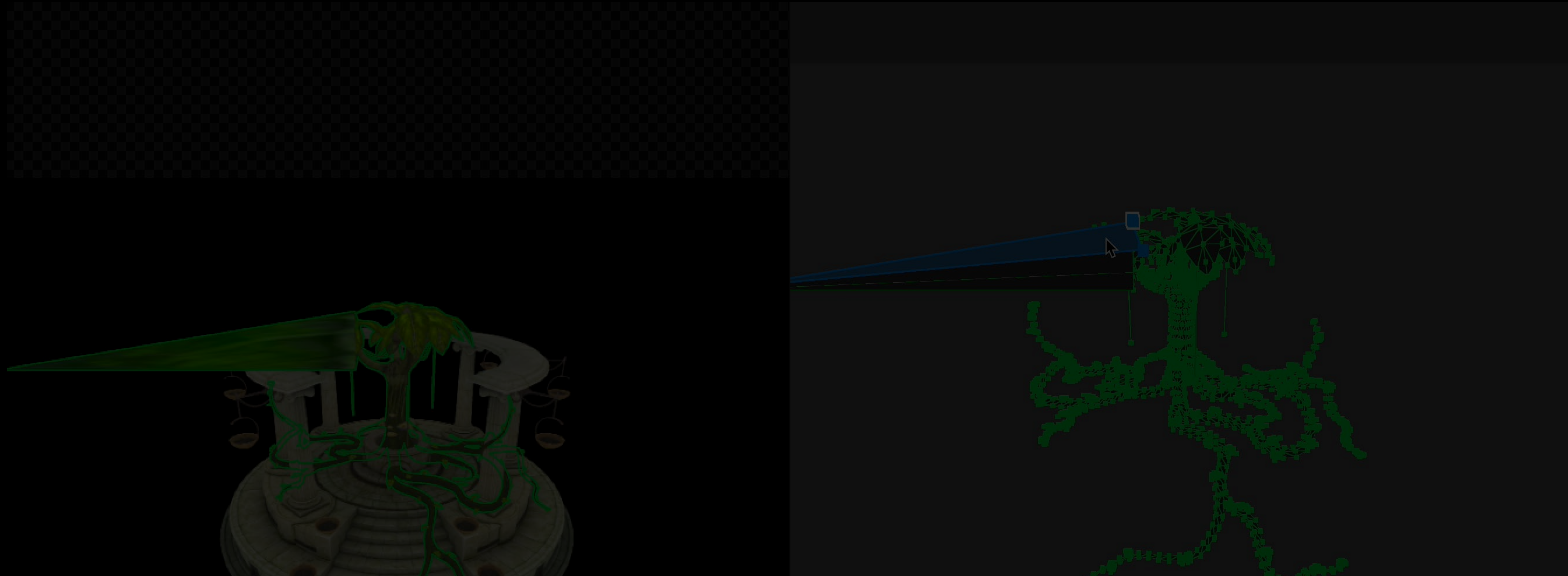
Visibly Wrong Triangles



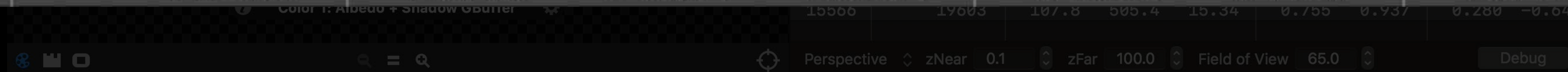
Index	in - uint Vertex	in - float3 position			in - float2 tex_coord		in - float2 norm	
15564	19614	87.38	482.3	16.96	0.682	0.884	0.420	-0.300
15565	19615	85.97	502.9	6.374	0.755	0.877	0.361	-0.600
15566	19603	107.8	505.4	15.34	0.755	0.937	0.280	-0.640



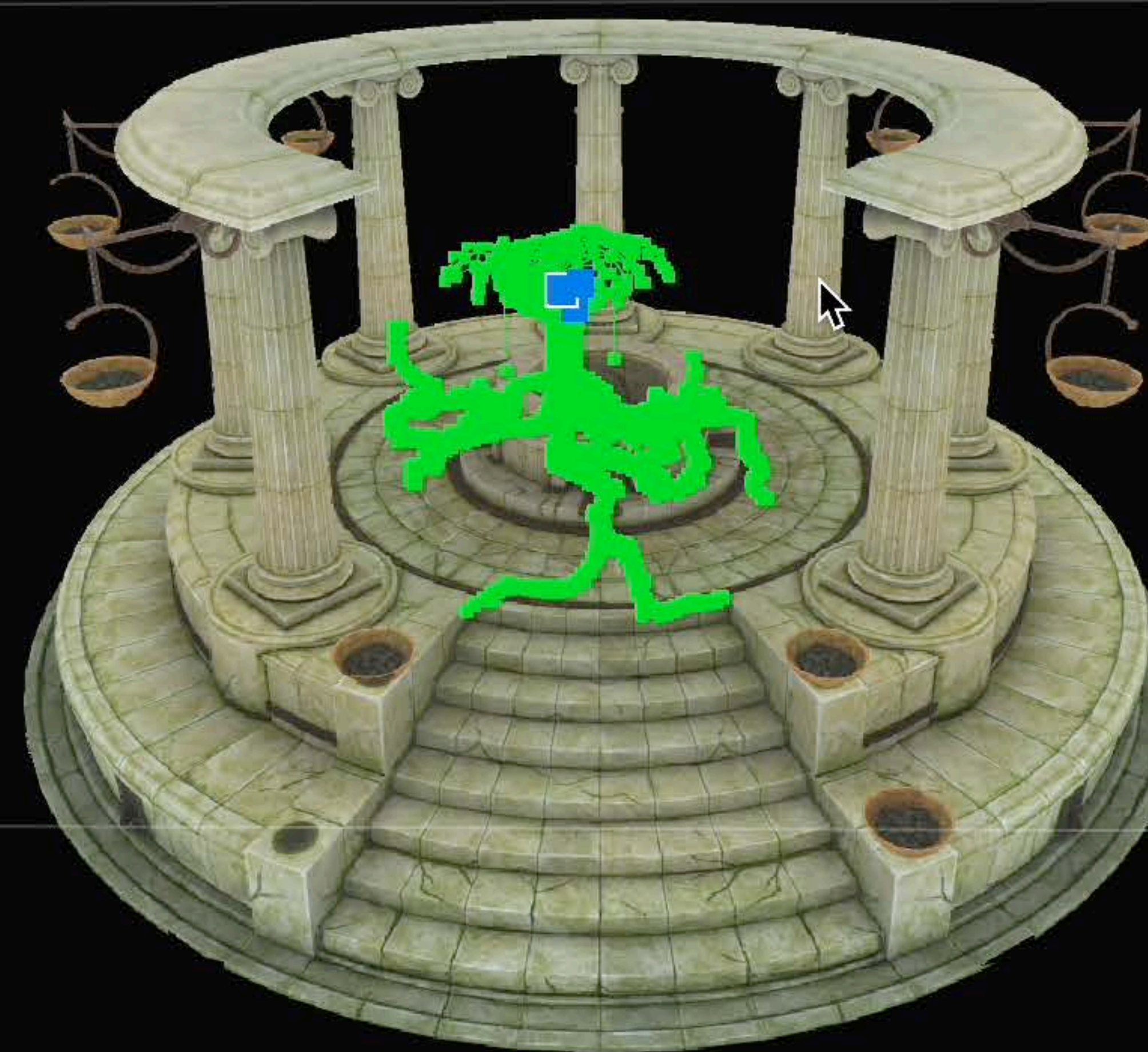
Visibly Wrong Triangles



Index	in - uint Vertex	in - float3 position			in - float2 tex_coord		in - float2 norm	
15564	19614	87.38	482.3	16.96	0.682	0.884	0.420	-0.300
15565	19615	85.97	502.9	6.374	0.755	0.877	0.361	-0.604
15566	19603	107.8	505.4	15.34	0.755	0.937	0.280	-0.647



Out of Frustum Objects

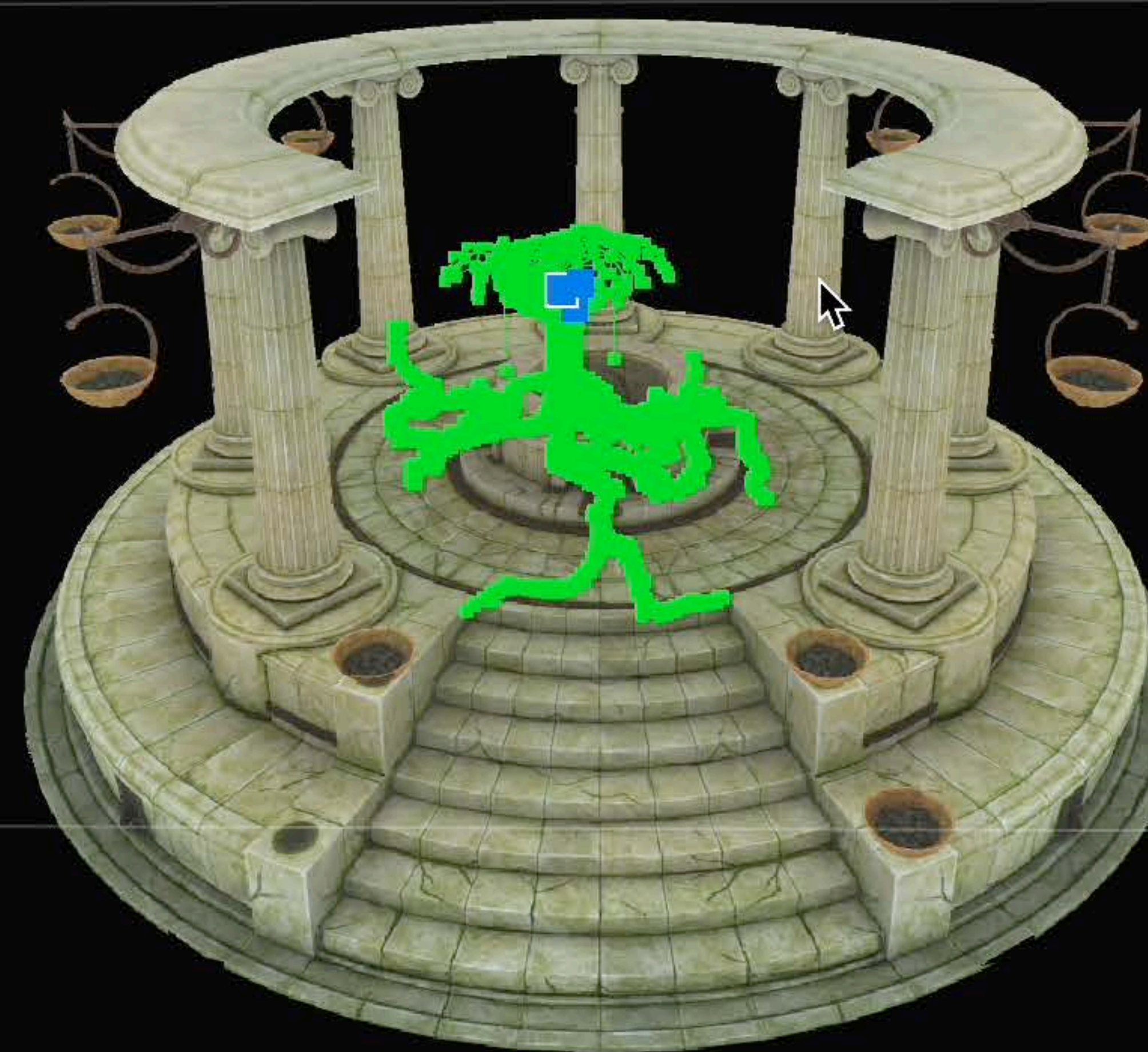


Color 1: Albedo + Shadow GBuffer

Index	in - uint Vertex	in - float3 position			in - float2 tex_coord		in - half4 normal		
14739	19346	-26.32	511.6	227.8	0.660	0.816	-0.029	0.582	0
14740	19328	-2.610	538.5	205.0	0.573	0.776	0.130	0.712	0
14741	19326	-37.10	549.4	200.3	0.573	0.852	-0.073	0.724	0

Perspective zNear 1.0 zFar 50.0 Field of View 65.0 Debug

Out of Frustum Objects

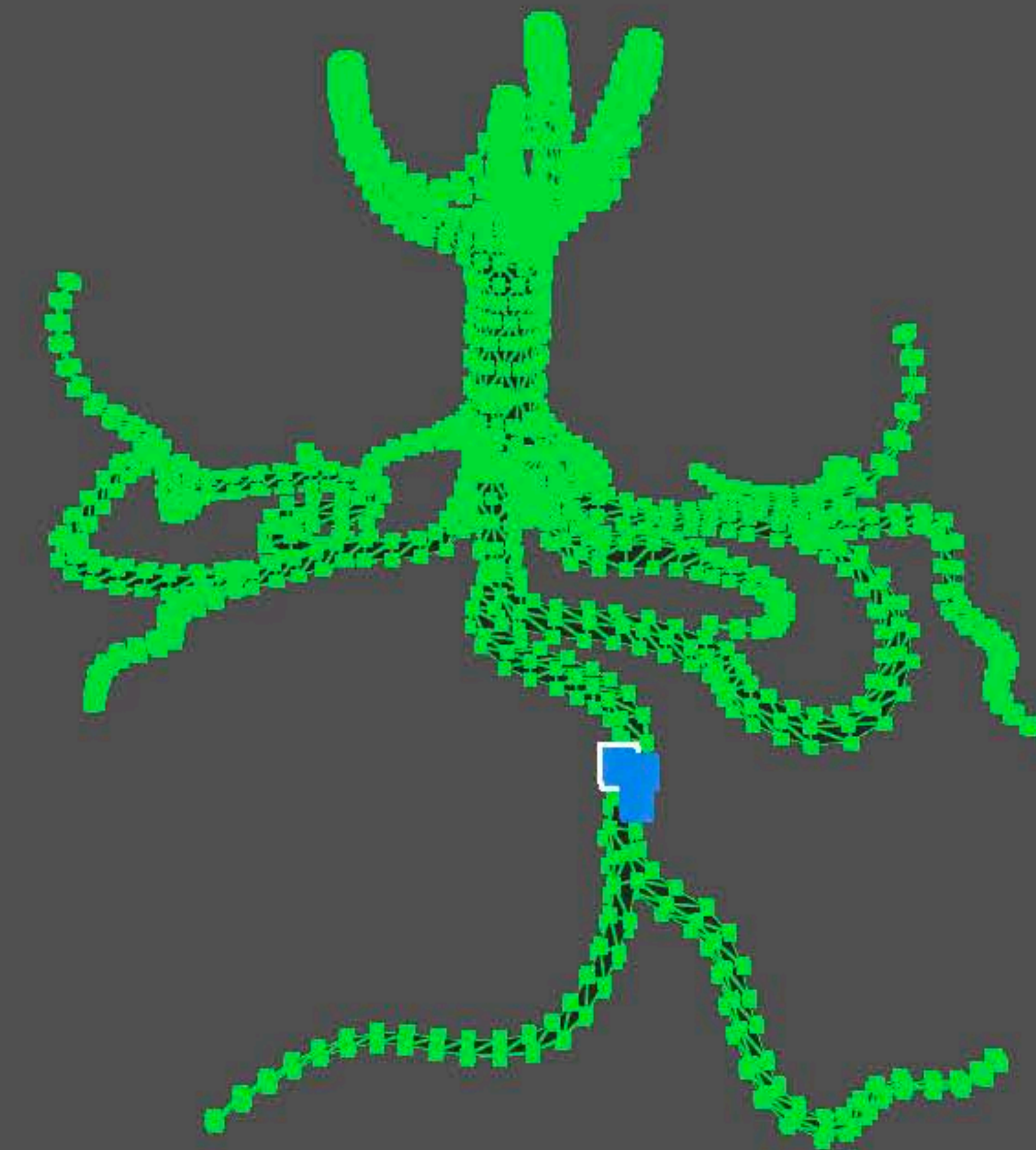


Color 1: Albedo + Shadow GBuffer

Index	in - uint Vertex	in - float3 position			in - float2 tex_coord		in - half4 normal		
14739	19346	-26.32	511.6	227.8	0.660	0.816	-0.029	0.582	0
14740	19328	-2.610	538.5	205.0	0.573	0.776	0.130	0.712	0
14741	19326	-37.10	549.4	200.3	0.573	0.852	-0.073	0.724	0

Perspective zNear 1.0 zFar 50.0 Field of View 65.0 Debug

Missing Triangles

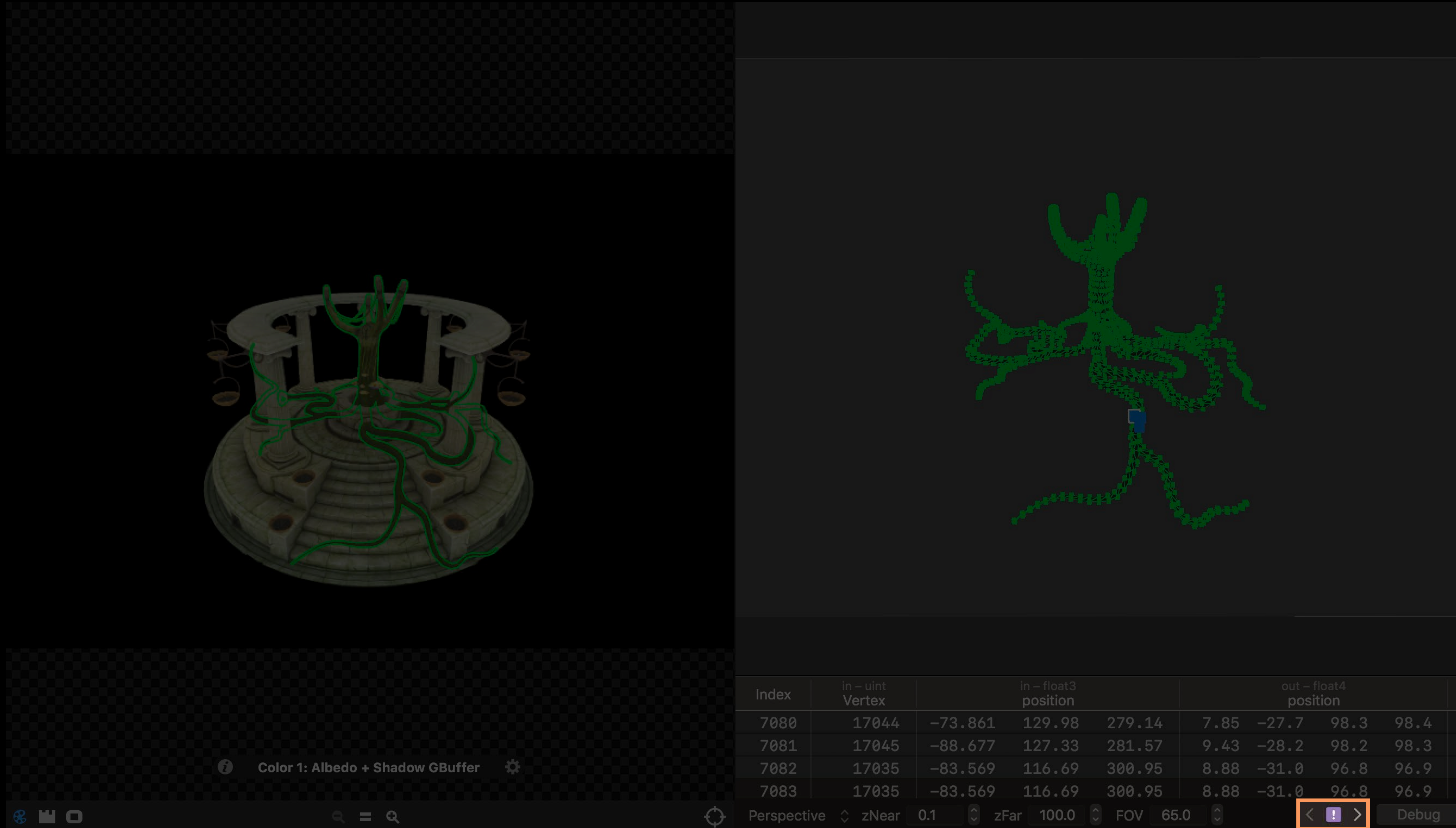


Color 1: Albedo + Shadow GBuffer

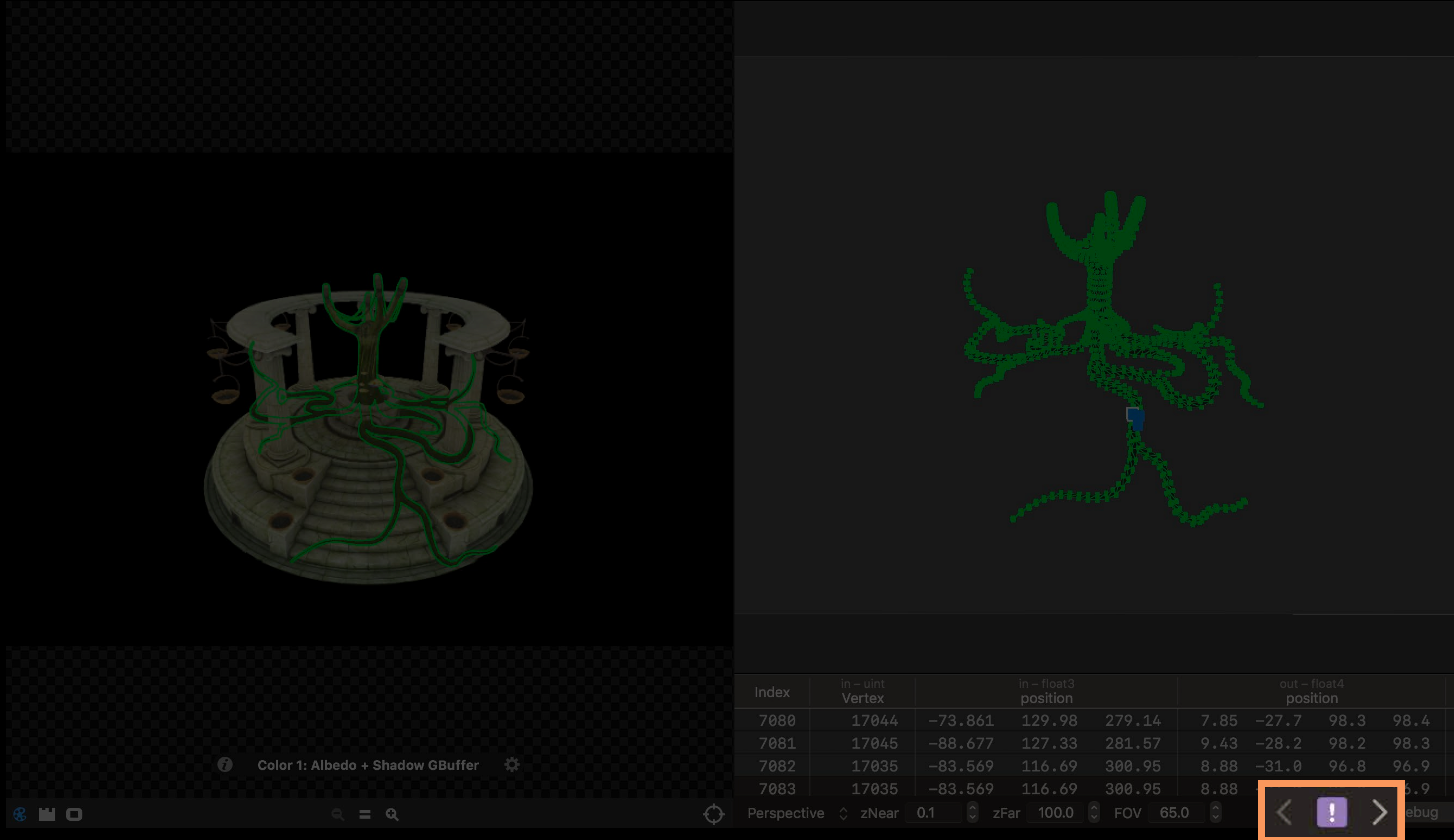
Index	in - uint Vertex	in - float3 position			out - float4 position			
7080	17044	-73.861	129.98	279.14	7.85	-27.7	98.3	98.4
7081	17045	-88.677	127.33	281.57	9.43	-28.2	98.2	98.3
7082	17035	-83.569	116.69	300.95	8.88	-31.0	96.8	96.9
7083	17035	-83.569	116.69	300.95	8.88	-31.0	96.8	96.9

Perspective zNear 0.1 zFar 100.0 FOV 65.0 Debug

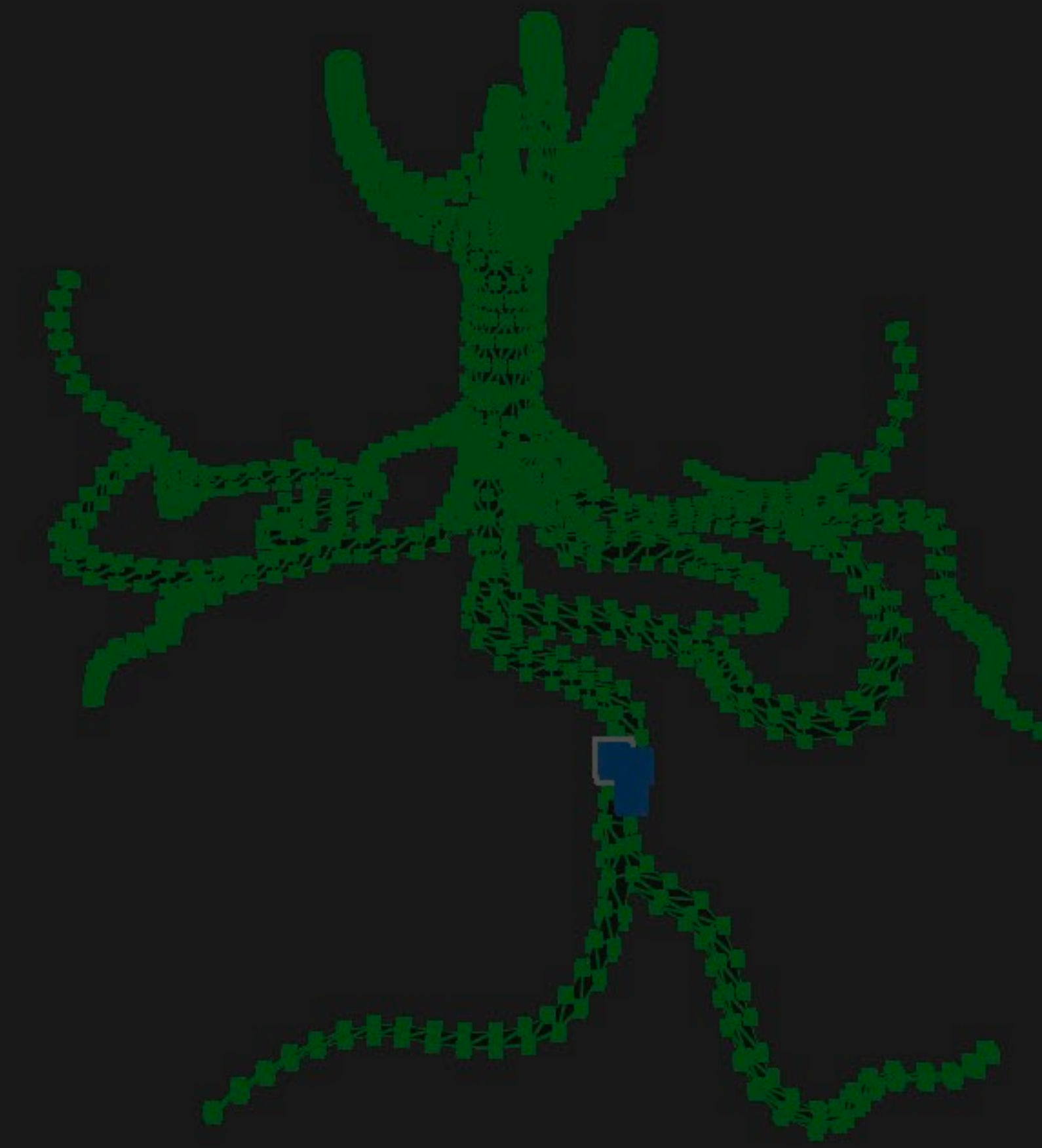
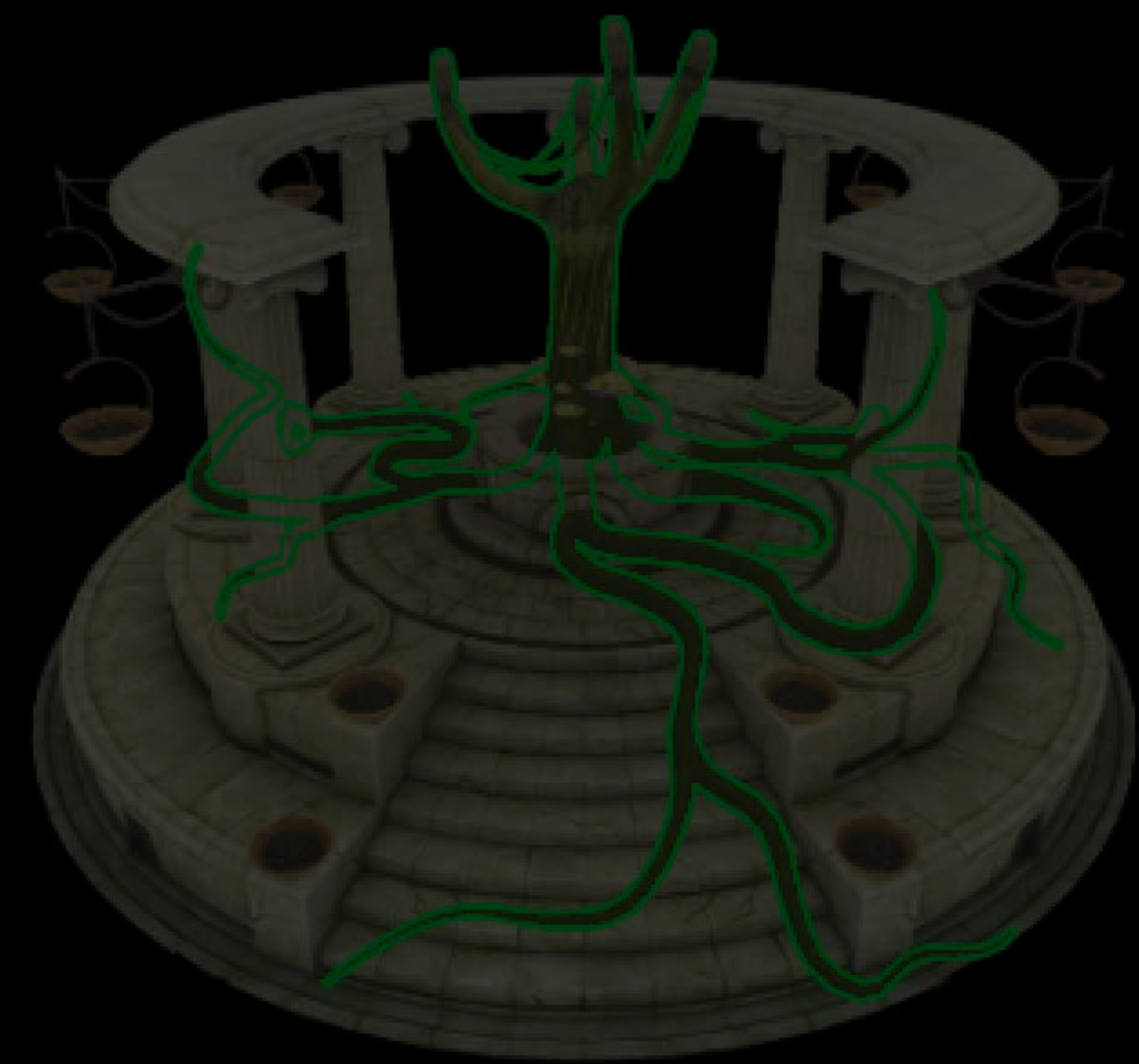
Missing Triangles



Missing Triangles



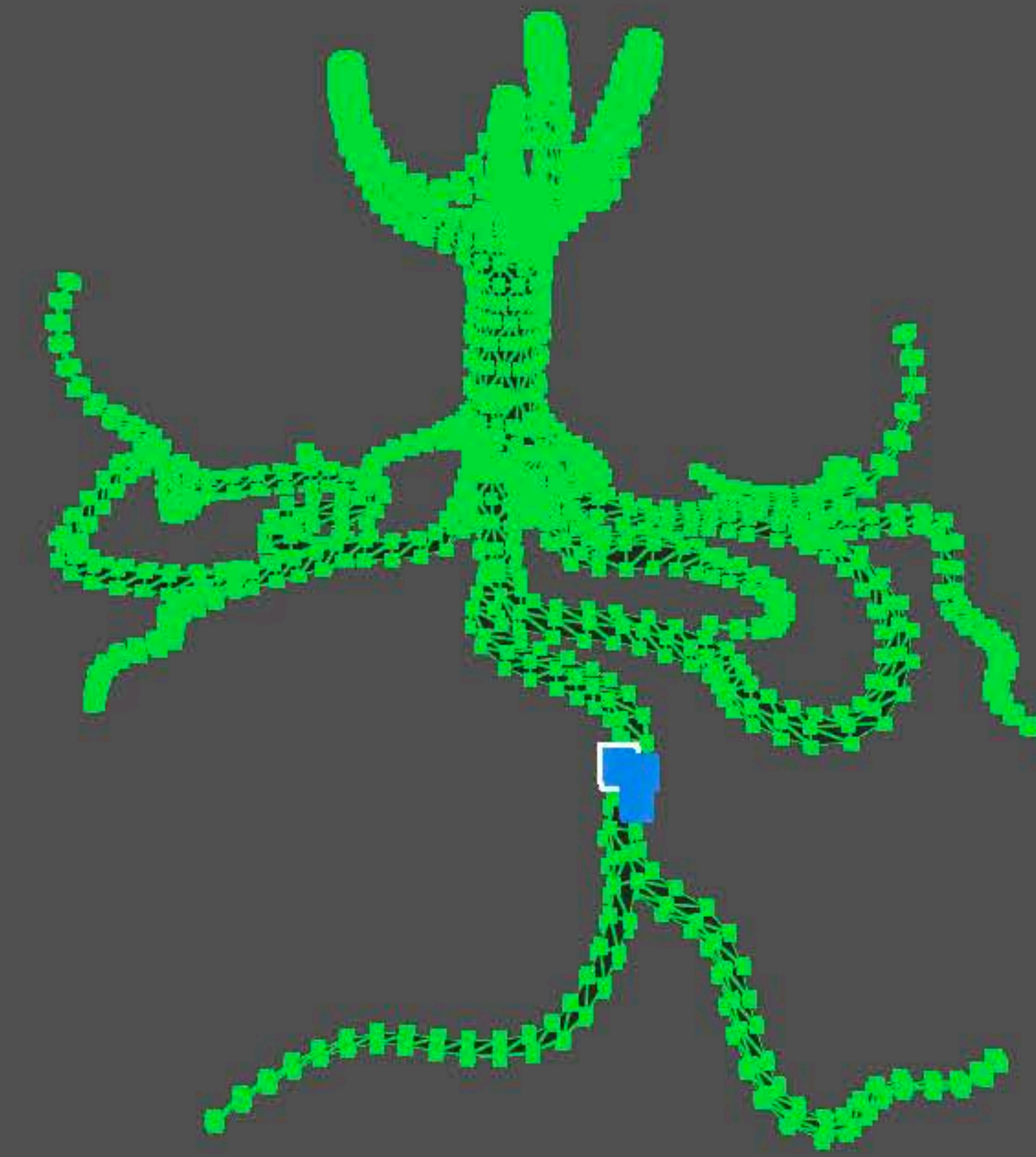
Missing Triangles



Index	in - uint Vertex	in - float3 position			out - float4 position			
7080	17044	-73.861	129.98	279.14	7.85	-27.7	98.3	98.4
7081	17045	-88.677	127.33	281.57	9.43	-28.2	98.2	98.3
7082	17035	-83.569	116.69	300.95	8.88	-31.0	96.8	96.9
7083	17035	-83.569	116.69	300.95	8.88	-31.0	96.8	96.9

Perspective zNear 0.1 zFar 100.0 FOV 65.0 !> Debug

Missing Triangles



Color 1: Albedo + Shadow GBuffer

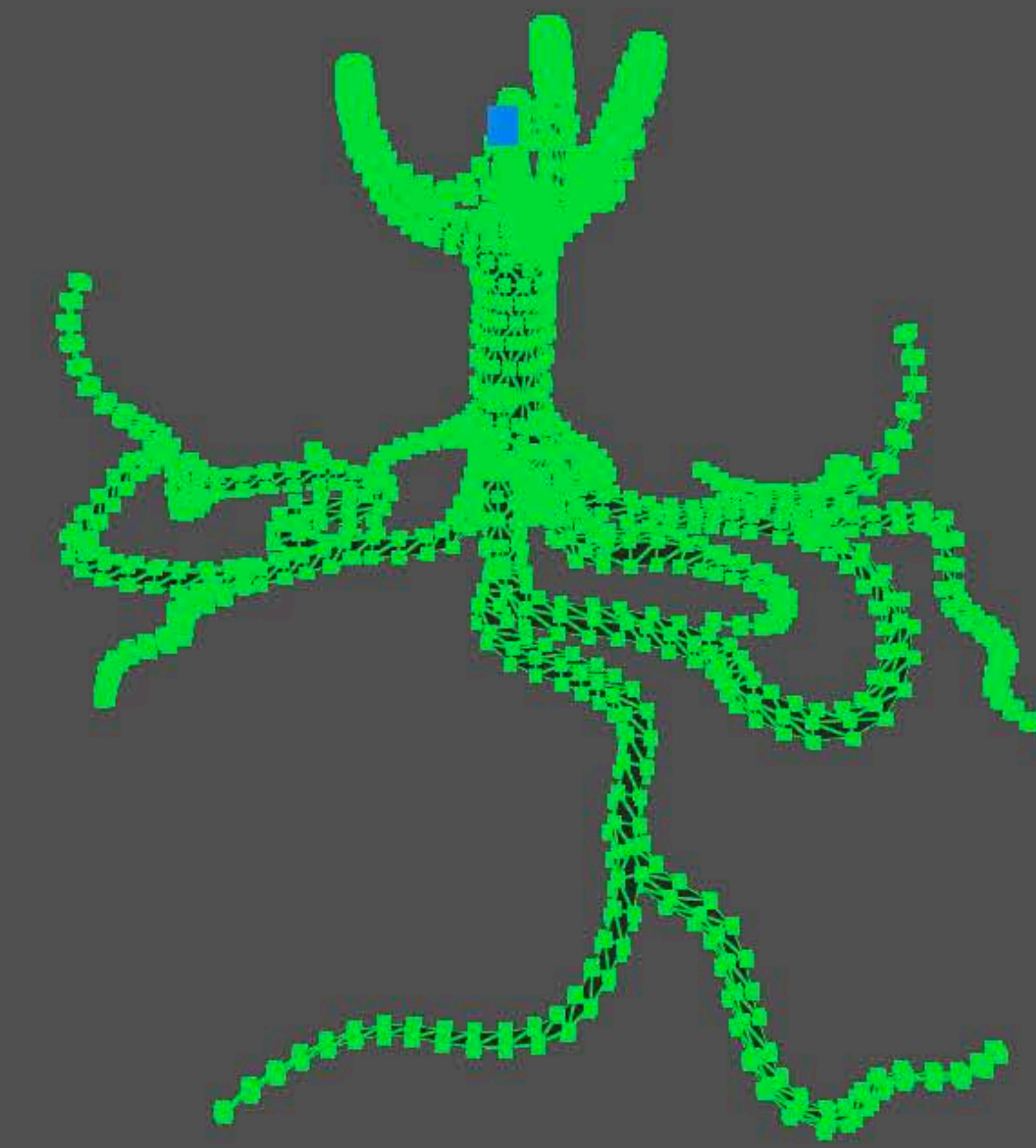
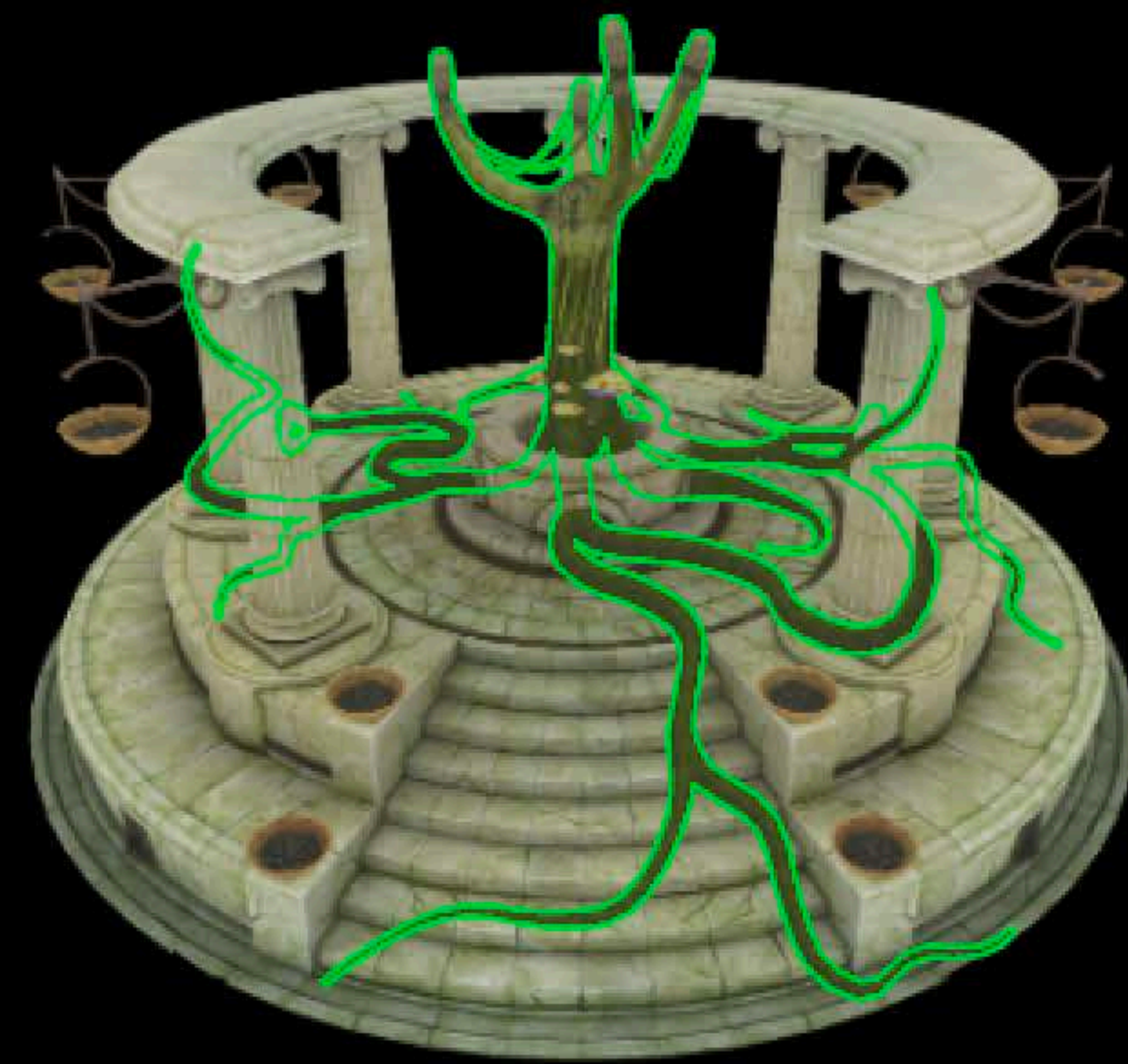
Index	in - uint Vertex	in - float3 position			
7080	17044	-73.861	129.98	279.14	7.85
7081	17045	-88.677	127.33	281.57	9.43
7082	17035	-83.569	116.69	300.95	8.88
7083	17035	-83.569	116.69	300.95	8.88

- Degenerates
- Triangle at Index 14250
 - Triangle at Index 14253
 - Triangle at Index 14262
 - Triangle at Index 14268
 - Triangle at Index 14271
 - Triangle at Index 14274
 - Triangle at Index 14277
 - Triangle at Index 14280
 - Triangle at Index 14283
 - Triangle at Index 14289

Perspective zNear 0.1 zFar 100.0 FOV 65.0

Debug

Missing Triangles

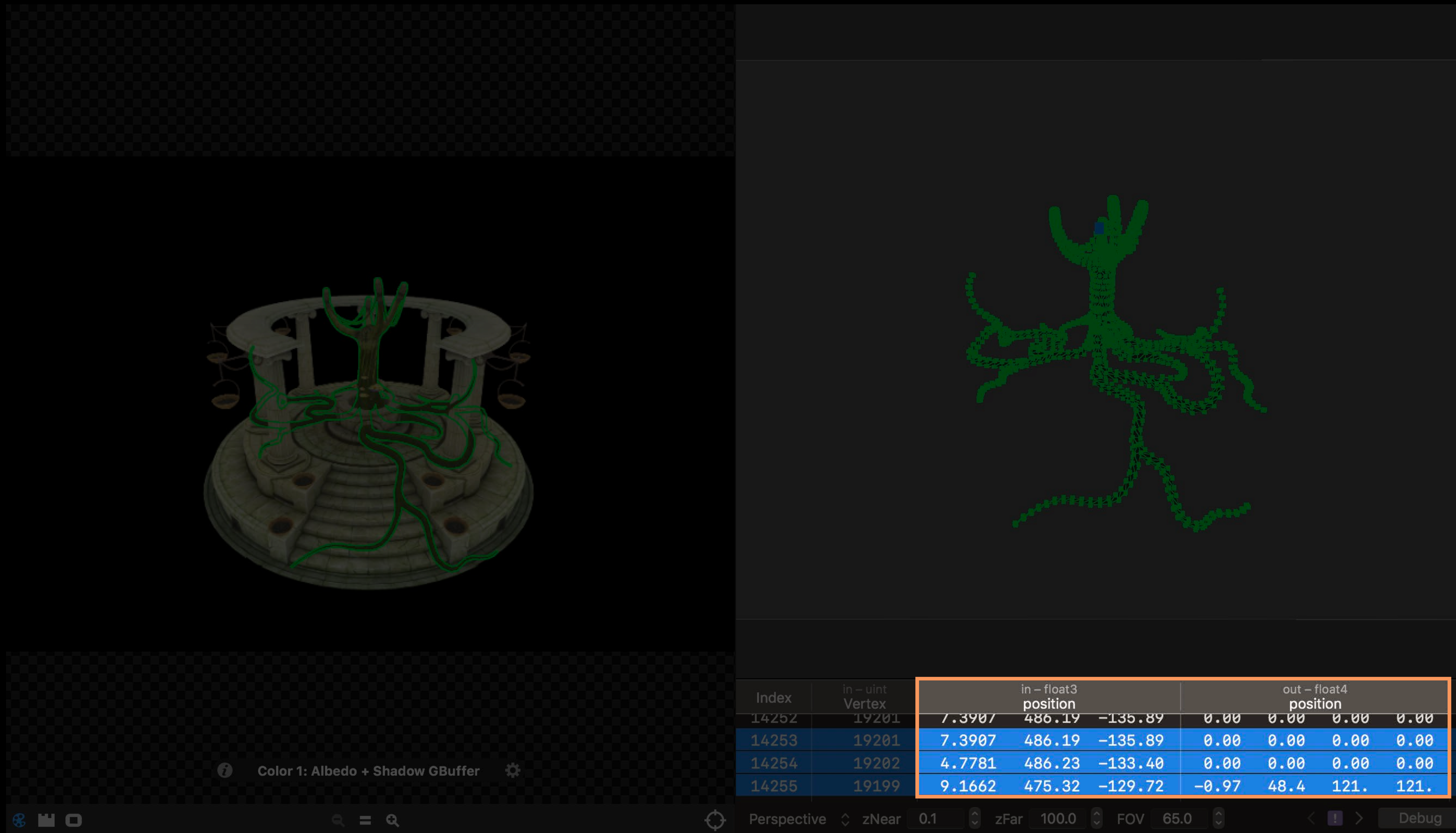


Color 1: Albedo + Shadow GBuffer

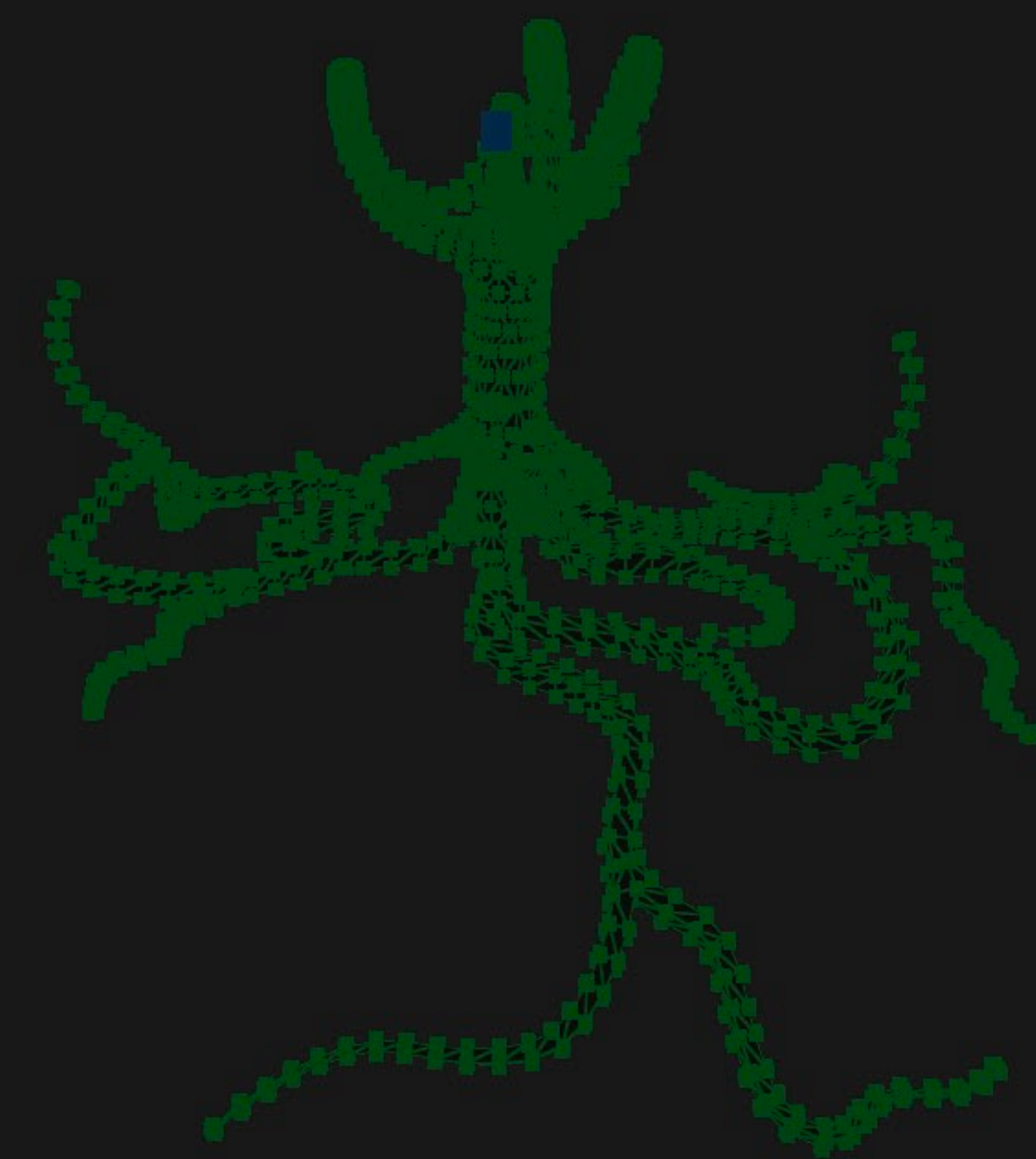
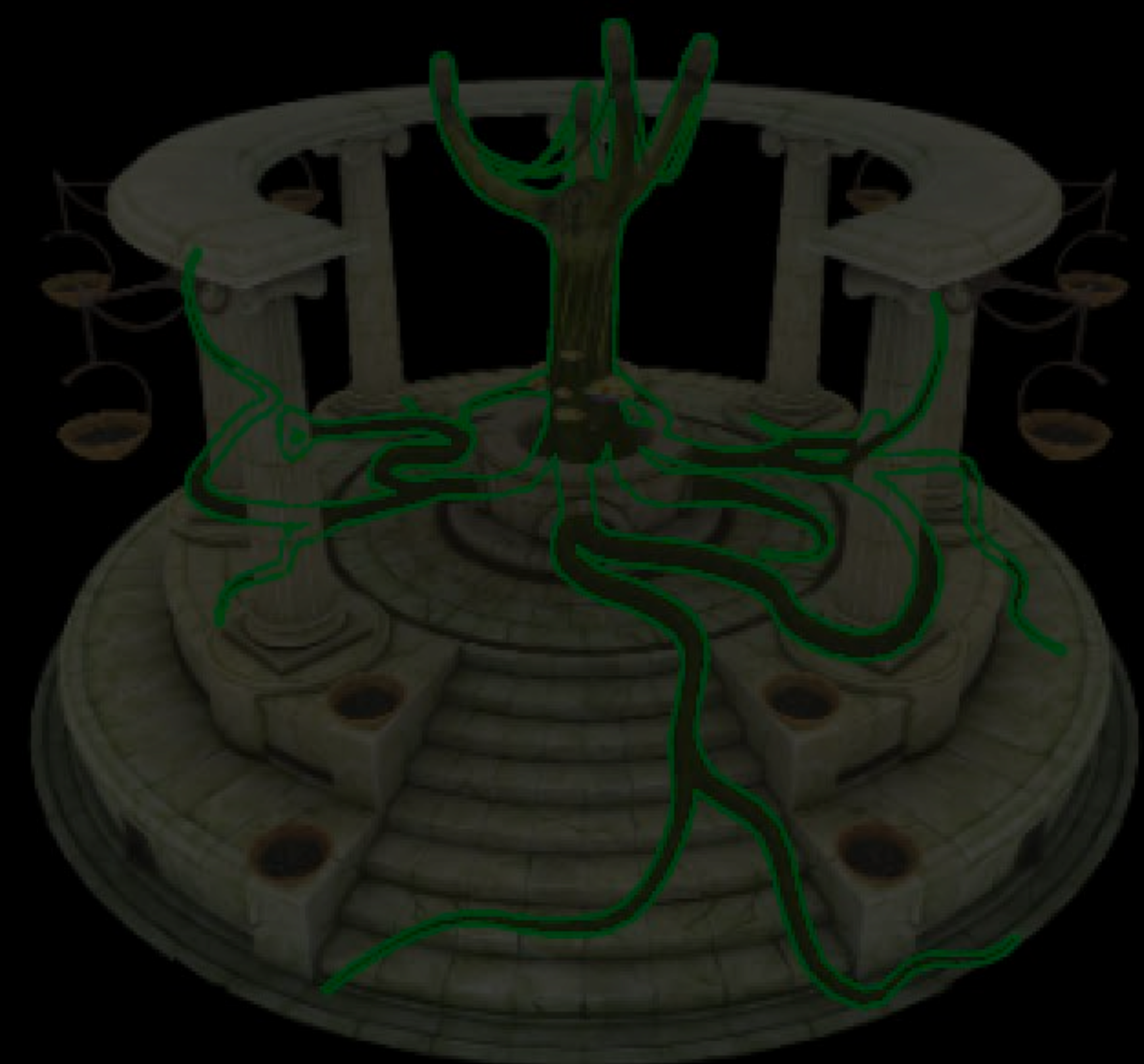
Index	in - uint Vertex	in - float3 position			out - float4 position			
14252	19201	7.3907	486.19	-135.89	0.00	0.00	0.00	0.00
14253	19201	7.3907	486.19	-135.89	0.00	0.00	0.00	0.00
14254	19202	4.7781	486.23	-133.40	0.00	0.00	0.00	0.00
14255	19199	9.1662	475.32	-129.72	-0.97	48.4	121.	121.

Perspective zNear 0.1 zFar 100.0 FOV 65.0 Debug

Missing Triangles



Missing Triangles



	in - float3 position			out - float4 position			
7.3907	486.19	-135.89	0.00	0.00	0.00	0.00	
7.3907	486.19	-135.89	0.00	0.00	0.00	0.00	
4.7781	486.23	-133.40	0.00	0.00	0.00	0.00	
9.1662	475.32	-129.72	-0.97	48.4	121.	121.	

Debugging Shaders with Shader Debugger

Shader Debugging

Math heavy code

Shader Debugging

Math heavy code

Highly parallel

Shader Debugging

Math heavy code

Highly parallel

Unity's "Book of the Dead"

- ~10 million vertex shader invocations
- ~60 million pixels rendered



Deferred Lighting - Debugging GPU Frame

Deferred Lighting Captured GPU Frame

FPS: 61 FPS

GPU

X: 1,509.5
Y: 951.5
Z: 0.9991762042045593

fragment GBufferData gbuffer_fragment(ColorInOut...)

- constant AAPLUniforms &uniforms [[buffer(AA...
- texture2d<half> baseColorMap [[texture(AAP...
- texture2d<half> normalMap [[texture(AAPL...
- texture2d<half> specularMap [[texture(AAPL...
- depth2d<float> shadowMap [[texture(AAP...
- constexpr sampler linearSampler(mip_filter::linear,
- half4 base_color_sample = baseColorMap.sample(...
- half4 normal_sample = normalMap.sample(linearS...
- half specular_contrib = specularMap.sample(linear...
- GBufferData gBuffer;
- half3 tangent_normal = normalize((normal_sample...
- half3 eye_normal = (tangent_normal.x * in.tangent +
- eye_normal = normalize(eye_normal);
- constexpr sampler shadowSampler(coord::normali...
- float shadow_sample = shadowMap.sample_comp...
- gBuffer.albedo_specular = half4(base_color_sampl...
- gBuffer.normal_shadow = half4(eye_normal.xyz, s...
- gBuffer.depth = in.eye_position.z;
- gBuffer.albedo_specular = pow(gBuffer.albedo_sp...
- }

```
102 // Fill in on-chip geometry buffer data
103 GBufferData gBuffer;
104
105 // Calculate normal in eye space
106 half3 tangent_normal = normalize((normal_sample.xyz * 2.0) - 1.0);
107
108 half3 eye_normal = (tangent_normal.x * in.tangent +
    eye_normal = (half3) [0.4086914, -0.6098633, 0.28271484]
    Values: [Color image] Mask: [Binary image]
    Min Value [-1.15234375, -0.80029296875, -0.50048828125, 0.0]
    Max Value [0.81982421875, 1.0830078125, 0.9658203125, 0.0]
    tangent_normal.y * in.bitangent +
    tangent_normal.z * in.normal);
109
110
111
112 eye_normal = normalize(eye_normal);
113
114 constexpr sampler shadowSampler(coord::normalized,
115     filter::linear,
116     mip_filter::none,
117     address::clamp_to_edge,
118     compare_func::less);
119
120 // Compare the depth value in the shadow map to the depth value of the
121 // fragment in the sun's.
122 // frame of reference. If the sample is occluded, it will be zero.
123 // #-code-listing(occlusionTest)
124 float shadow_sample = shadowMap.sample_compare(shadowSampler,
    in.shadow_coord.xy, in.shadow_coord.z);
125 // #-end-code-listing
```

gBuffer = [[n/a, n/a, n/a, n/a],

tangent_normal = [0.099365,

eye_normal = [0.4086914, -0.

eye_normal = [0.51904297, -

shadowSampler = [0x7c2481

shadow_sample = 0.0

Color 1: Albedo + Shadow GBuffer

in = (ColorInOut) [[1509.5, 951.5, 0.9991762, 0.00873627], [0.47408918, 0.96787506], [0.60012203, 0.4589786, 0.3803352], [1.8502556, 3.881843, 114.46534], [0.82421875, 0.13952637, 0.46484375], [-0.7294922, 0.1385498, 0.62695...

uniforms = (AAPLUniforms) [[[[1.0630283, 0.0, 0.0, 0.0], [0.0, 1.5696856, 0.0, 0.0], [0.0, 0.0, 1.00005, 1.0], [0.0, 0.0, -0.10000499, 0.0]]], [[[0.9407085, 0.0, 0.0, 0.0], [0.0, 0.6370701, 0.0, 0.0], [0.0, 0.0, 0.0, -9.999498], [0.0, 0.0, 0.9999999, ...

baseColorMap = (texture2d<half, metal::access::sample, void>) [0x100a161a0]

normalMap = (texture2d<half, metal::access::sample, void>) [0x100908ef0]

specularMap = (texture2d<half, metal::access::sample, void>) [0x10081a8f0]

shadowMap = (depth2d<float, metal::access::sample, void>) [0x17f804032d1d9557]

linearSampler = (sampler) [0x782481bffc002034]

base_color_sample = (half4) [0.32348633, 0.1472168, 0.41210938, 1.0]

normal_sample = (half4) [0.53222656, 0.35766602, 0.7895508, 1.0]

specular_contrib = (half) 0.62402344

gBuffer = (GBufferData) [[n/a, n/a, n/a, n/a] [n/a, n/a, n/a, n/a] [n/a, n/a, n/a, n/a] [n/a, n/a, n/a, n/a]

Deferred Lighting - Debugging GPU Frame

Deferred Lighting.gputrace > Fragment Render Pipeline — gbuffer_fragment > gbuffer_fragment

Automatic Attachments

Deferred Lighting Captured GPU Frame

FPS 61 FPS

GPU

X: 1,509.5
Y: 951.5
Z: 0.9991762042045593

fragment GBufferData gbuffer_fragment(ColorInOut...)

- constant AAPLUniforms & uniforms [[buffer(AA...
- texture2d<half> baseColorMap [[texture(AAP...
- texture2d<half> normalMap [[texture(AAPL...
- texture2d<half> specularMap [[texture(AAPL...
- depth2d<float> shadowMap [[texture(AAP...
- constexpr sampler linearSampler(mip_filter::linear,
- half4 base_color_sample = baseColorMap.sample(...
- half4 normal_sample = normalMap.sample(linearS...
- half specular_contrib = specularMap.sample(linear...
- GBufferData gBuffer;
- half3 tangent_normal = normalize((normal_sample...
- half3 eye_normal = (tangent_normal.x * in.tangent +
- eye_normal = normalize(eye_normal);
- constexpr sampler shadowSampler(coord::normali...
- float shadow_sample = shadowMap.sample_comp...
- gBuffer.albedo_specular = half4(base_color_sampl...
- gBuffer.normal_shadow = half4(eye_normal.xyz, s...
- gBuffer.depth = in.eye_position.z;
- gBuffer.albedo_specular = pow(gBuffer.albedo_sp...
- }

```
// Fill in on-chip geometry buffer data
GBufferData gBuffer;

// Calculate normal in eye space
half3 tangent_normal = normalize((normal_sample.xyz * 2.0) - 1.0);
half3 eye_normal = (tangent_normal.x * in.tangent +
    tangent_normal.y * in.bitangent +
    tangent_normal.z * in.normal);
eye_normal = normalize(eye_normal);

constexpr sampler shadowSampler(coord::normalized,
    filter::linear,
    mip_filter::none,
    address::clamp_to_edge,
    compare_func::less);

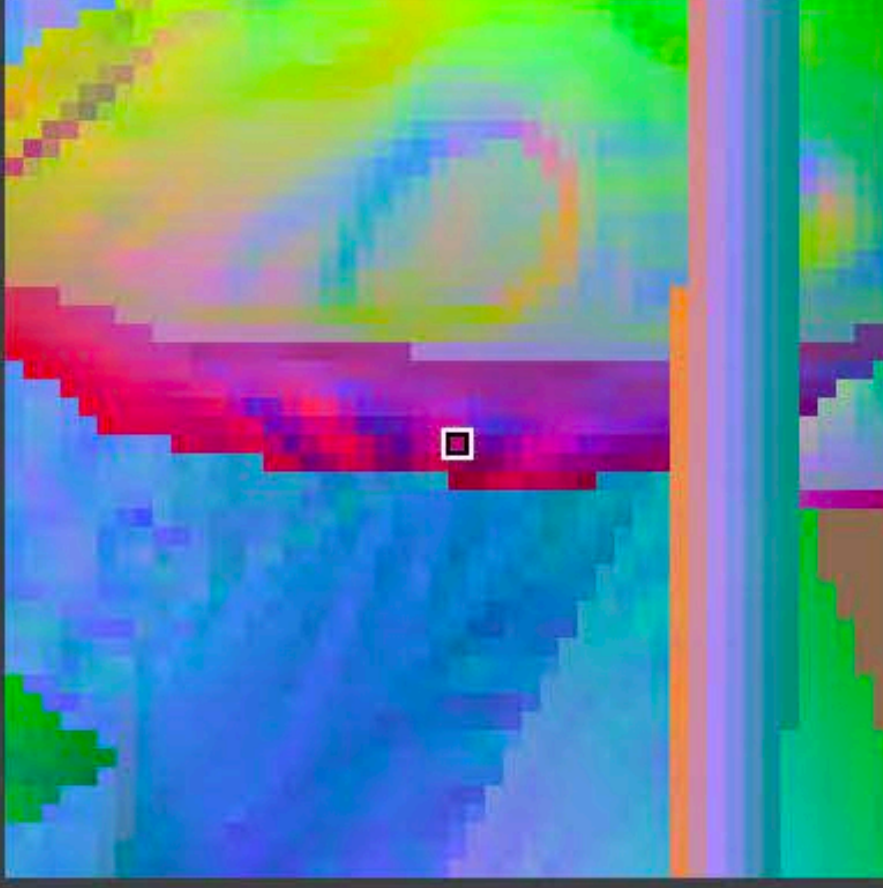
// Compare the depth value in the shadow map to the depth value of the
// fragment in the sun's
// frame of reference. If the sample is occluded, it will be zero.
//--code-listing(occlusionTest)
float shadow_sample = shadowMap.sample_compare(shadowSampler,
    in.shadow_coord.xy, in.shadow_coord.z);
//--end-code-listing
```

gBuffer = [[n/a, n/a, n/a, n/a],

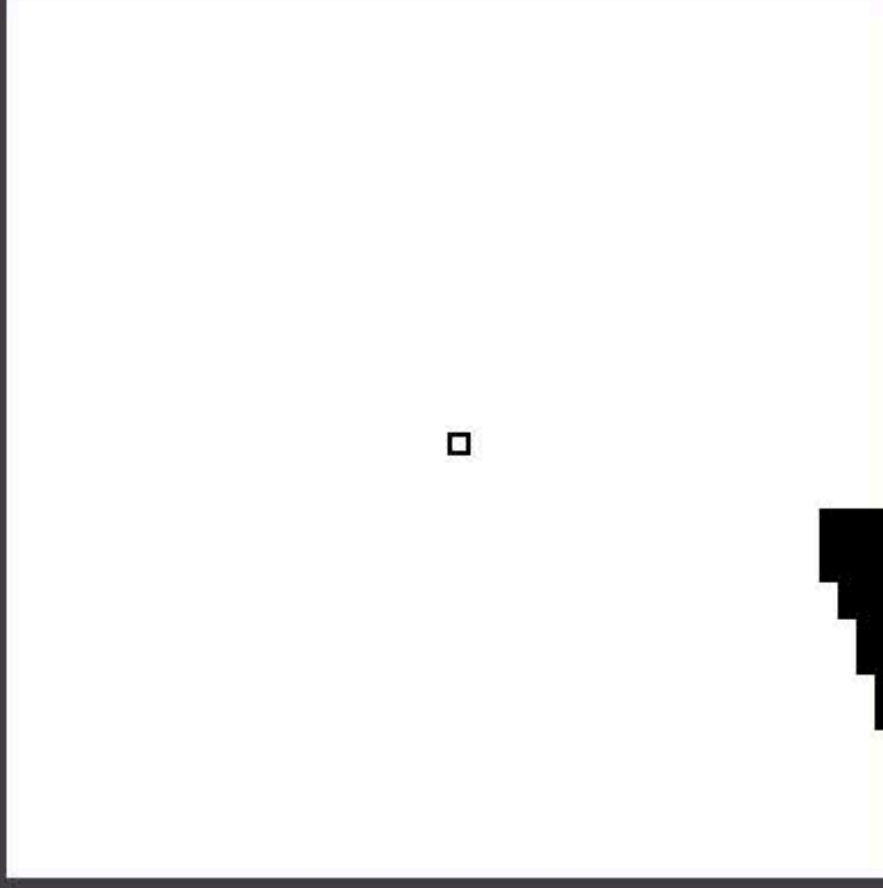
tangent_normal = [0.099365,

eye_normal = [0.4086914, -0.

Values



Mask



Min Value [-1.15234375, -0.80029296875, -0.50048828125, 0.0]

Max Value [0.81982421875, 1.0830078125, 0.9658203125, 0.0]

tangent_normal.y * in.bitangent +
tangent_normal.z * in.normal);

eye_normal = [0.51904297, -

shadowSampler = [0x7c2481

shadow_sample = 0.0

Color 1: Albedo + Shadow GBuffer

in = (ColorInOut) [[1509.5, 951.5, 0.9991762, 0.00873627], [0.47408918, 0.96787506], [0.60012203, 0.4589786, 0.3803352], [1.8502556, 3.881843, 114.46534], [0.82421875, 0.13952637, 0.46484375], [-0.7294922, 0.1385498, 0.62695...

uniforms = (AAPLUniforms) [[[[[1.0630283, 0.0, 0.0, 0.0], [0.0, 1.5696856, 0.0, 0.0], [0.0, 0.0, 1.00005, 1.0], [0.0, 0.0, -0.10000499, 0.0]]], [[[0.9407085, 0.0, 0.0, 0.0], [0.0, 0.6370701, 0.0, 0.0], [0.0, 0.0, 0.0, -9.999498], [0.0, 0.0, 0.9999999, ...

baseColorMap = (texture2d<half, metal::access::sample, void>) [0x100a161a0]

normalMap = (texture2d<half, metal::access::sample, void>) [0x100908ef0]

specularMap = (texture2d<half, metal::access::sample, void>) [0x10081a8f0]

shadowMap = (depth2d<float, metal::access::sample, void>) [0x17f804032d1d9557]

linearSampler = (sampler) [0x782481bffc002034]

base_color_sample = (half4) [0.32348633, 0.1472168, 0.41210938, 1.0]

normal_sample = (half4) [0.53222656, 0.35766602, 0.7895508, 1.0]

specular_contrib = (half) 0.62402344

gBuffer = (GBufferData) [[n/a n/a n/a n/a] [n/a n/a n/a n/a] [n/a n/a n/a n/a] [n/a n/a n/a n/a] n/a]

Shader Debugger

NEW

New tool for debugging Metal shaders

Rich variable visualization across thousands of threads

Real data from GPU

```
// Fill in on-chip geometry buffer data
GBufferData gBuffer;

// Calculate normal in eye space
half3 tangent_normal = normalize((normal_sample.xyz * 2.0) - 1.0);

half3 eye_normal = (tangent_normal.x * in.tangent +
eye_normal = (half3) [0.4086914, -0.6098633, 0.28271484]
tangent_normal.y * in.bitangent +
tangent_normal.z * in.normal);

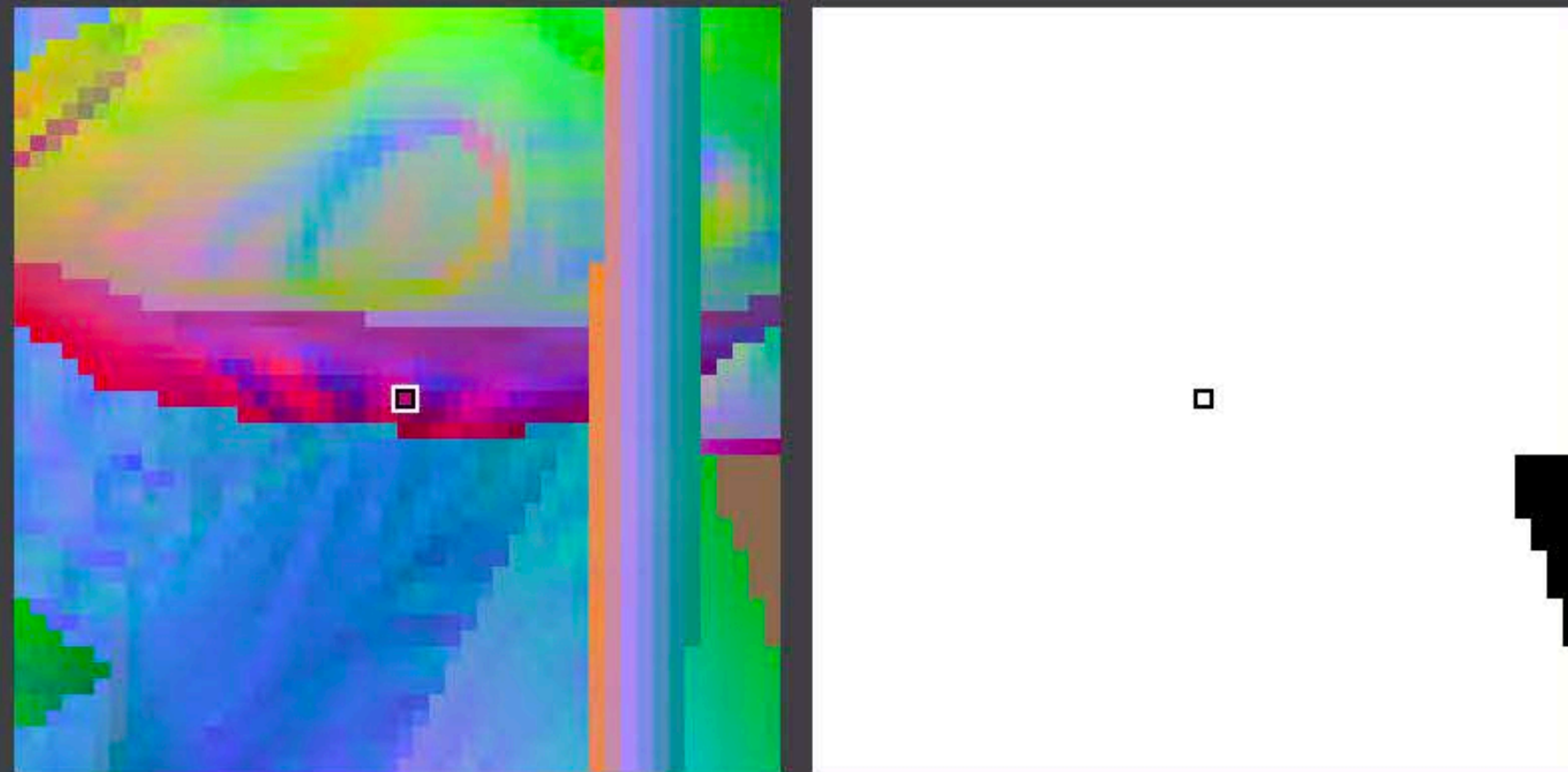
eye_normal = normalize(eye_normal);
```

gBuffer = [[n/a, n/a, n/a, n/a],

tangent_normal = [0.099365,

eye_normal = [0.4086914, -0

Values Mask



Min Value [-1.15234375, -0.80029296875, -0.50048828125, 0.0]

Max Value [0.81982421875, 1.0830078125, 0.9658203125, 0.0]

eye_normal = [0.51904297, -

Shader Debugger

NEW

New tool for debugging Metal shaders

Rich variable visualization across thousands of threads

Real data from GPU

Flexible stepping

```
// Fill in on-chip geometry buffer data
GBufferData gBuffer;

// Calculate normal in eye space
half3 tangent_normal = normalize((normal_sample.xyz * 2.0) - 1.0);

half3 eye_normal = (tangent_normal.x * in.tangent +
eye_normal = (half3) [0.4086914, -0.6098633, 0.28271484]
tangent_normal.y * in.bitangent +
tangent_normal.z * in.normal);

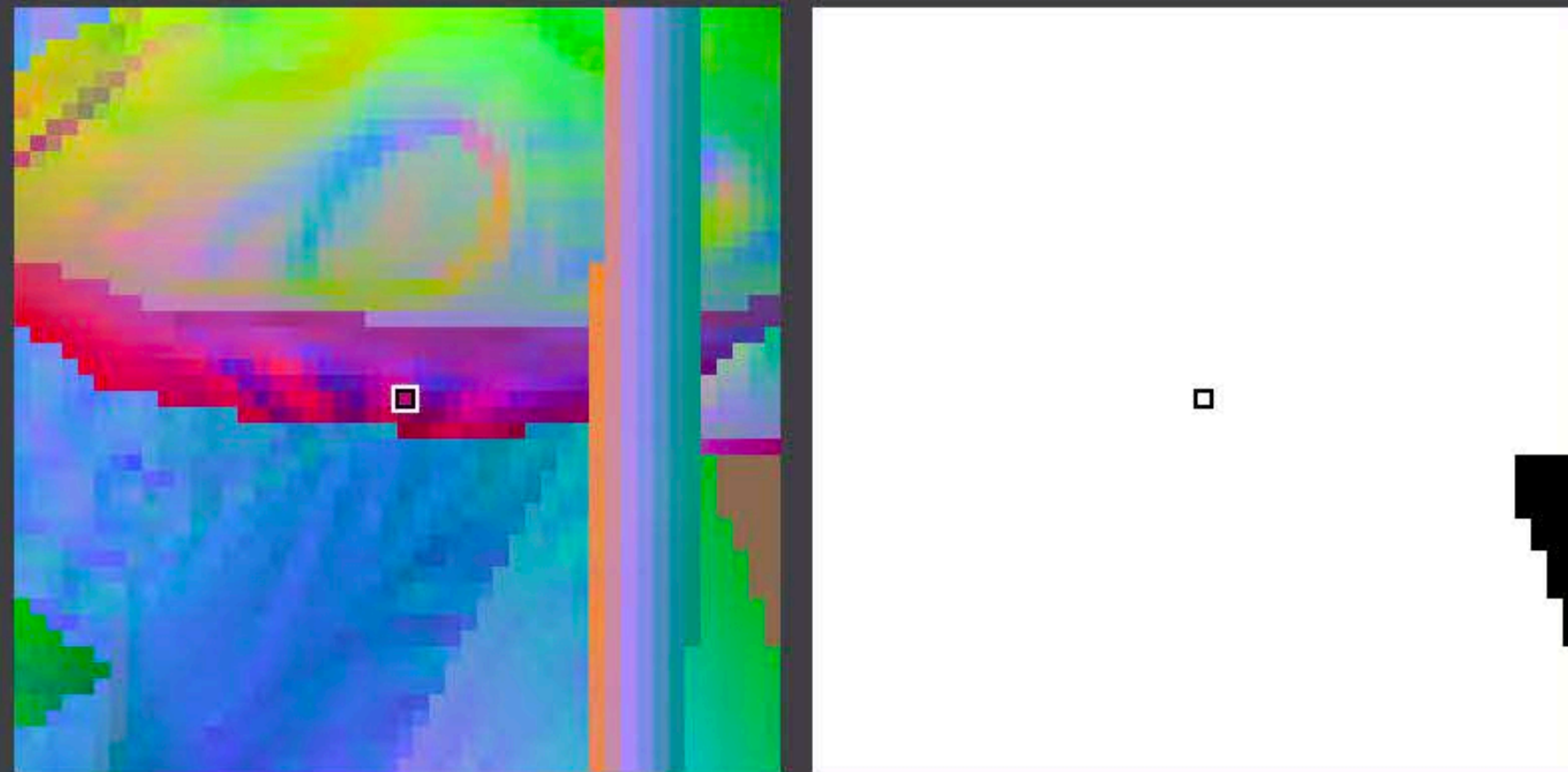
eye_normal = normalize(eye_normal);
```

gBuffer = [[n/a, n/a, n/a, n/a],

tangent_normal = [0.099365,

eye_normal = [0.4086914, -0

Values Mask



Min Value [-1.15234375, -0.80029296875, -0.50048828125, 0.0]

Max Value [0.81982421875, 1.0830078125, 0.9658203125, 0.0]

eye_normal = [0.51904297, -

Shader Debugger

NEW

New tool for debugging Metal shaders

Rich variable visualization across thousands of threads

Real data from GPU

Flexible stepping

Integrated into Metal Frame Debugger

```
// Fill in on-chip geometry buffer data
GBufferData gBuffer;

// Calculate normal in eye space
half3 tangent_normal = normalize((normal_sample.xyz * 2.0) - 1.0);

half3 eye_normal = (tangent_normal.x * in.tangent +
eye_normal = (half3) [0.4086914, -0.6098633, 0.28271484]
tangent_normal.y * in.bitangent +
tangent_normal.z * in.normal);

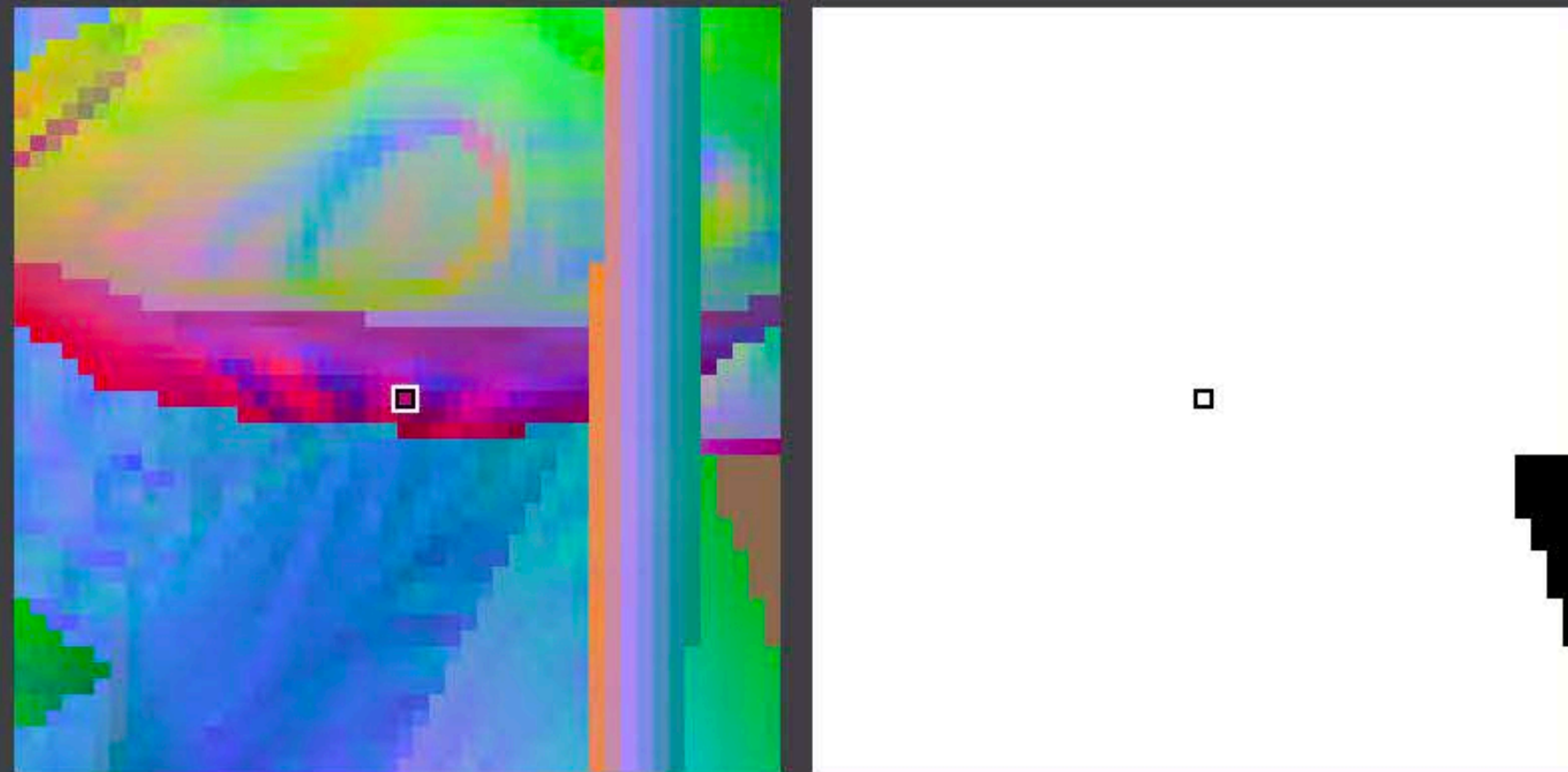
eye_normal = normalize(eye_normal);
```

gBuffer = [[n/a, n/a, n/a, n/a],

tangent_normal = [0.099365,

eye_normal = [0.4086914, -0

Values Mask



Min Value [-1.15234375, -0.80029296875, -0.50048828125, 0.0]

Max Value [0.81982421875, 1.0830078125, 0.9658203125, 0.0]

eye_normal = [0.51904297, -

Demo

Xavier Verguin Gonzalez

Starting the Shader Debugger

The screenshot displays the Shader Debugger interface for a scene. The left sidebar shows a tree view of the render command encoder, with the current command encoder selected. The main area is divided into three sections: a resource table, a render target view, and a render pass descriptor.

Label	Type	Size	Details
Vertex			
Buffer 0x100c17ed0	Buffer 0	56 bytes	Offset: 0x0 [vertices]
Buffer 0x100c17250	Buffer 1	5 KB	Offset: 0x0
UniformBuffer0	Buffer 2	640 bytes	Offset: 0x0 [uniforms]
LightData	Buffer 3	8 KB	Offset: 0x0 [light_data]
LightPositions0	Buffer 4	4 KB	Offset: 0x0 [light_positions]
Geometry	Post Vertex Trans...		
fairy_vertex	Vertex Function		Library 0x600001753640...
Fragment			
Sky Map	Texture 0	(Cube) 512 x 512	RGBA8Unorm_sRGB
Albedo + Shadow GBuffer	Texture 1	2968 x 2010	RGBA8Unorm_sRGB
Normal + Specular GBuffer	Texture 2	2968 x 2010	RGBA8Snorm
Depth GBuffer	Texture 3	2968 x 2010	R32Float
Fairy Map	Texture 4	64 x 64	RGBA8Unorm_sRGB [color...]
Texture 0x100b18e30	Color 0	2968 x 2010	BGRA8Unorm_sRGB
MTKView Depth Stencil	Depth	2968 x 2010	Depth32Float_Stencil8
MTKView Depth Stencil	Stencil	2968 x 2010	Depth32Float_Stencil8
UniformBuffer0	Buffer 2	640 bytes	Offset: 0x0
LightData	Buffer 3	8 KB	Offset: 0x0
LightPositions0	Buffer 4	4 KB	Offset: 0x0
fairy_fragment	Fragment Function		Library 0x600001753640...

The render target view shows two attachments: **Color 0** (a color image of the scene) and **Depth: MTKView Depth Stencil** (a depth and stencil image). The render pass descriptor at the bottom provides detailed information about the current render pass, including the visibility result buffer, OpenGL mode, render target array length, and the attachments used.

```
RenderPassDescriptor (MTLRenderPassDescriptor) 2968x2010 Color/Depth/Stencil
  Visibility Result Buffer 0x0 Does not exist
  OpenGLModeEnabled = (BOOL) NO
  RenderTargetArrayLength = (NSUInteger) 0
  Color 0 (MTLRenderPassAttachmentDescriptor) DontCare/Store Texture 0x100b18e30
  Depth (MTLRenderPassAttachmentDescriptor) Load/DontCare "MTKView Depth Stencil" (0x100a315d0)
  Stencil (MTLRenderPassAttachmentDescriptor) Load/DontCare "MTKView Depth Stencil" (0x100a315d0)
    "MTKView Depth Stencil" (0x100a315d0) (MTLTexture) 2968 x 2010, Depth32Float_Stencil8
      TextureLevel = (NSUInteger) 0
      TextureSlice = (NSUInteger) 0
      TextureDepthPlane = (NSUInteger) 0
```


8 KB Offset: 0x0

4 KB Offset: 0x0

Library 0x600001753640...



 Depth: MTKView Depth Stencil 



/Depth/Stencil

ure 0x100b18e30

"View Depth Stencil" (0x100a315d0)

"View Depth Stencil" (0x100a315d0)

x 2010, Depth32Float_Stencil8

8 KB Offset: 0x0

4 KB Offset: 0x0

Library 0x600001753640...



 Depth: MTKView Depth Stencil 



/Depth/Stencil

ure 0x100b18e30
"View Depth Stencil" (0x100a315d0)
"View Depth Stencil" (0x100a315d0)
x 2010, Depth32Float_Stencil8

8 KB Offset: 0x0

4 KB Offset: 0x0

Library 0x600001753640...

Vertex

Fragment



/Depth/Stencil

ure 0x100b18e30

View Depth Stencil" (0x100a315d0)

View Depth Stencil" (0x100a315d0)

x 2010, Depth32Float_Stencil8



1,630

987

Debug



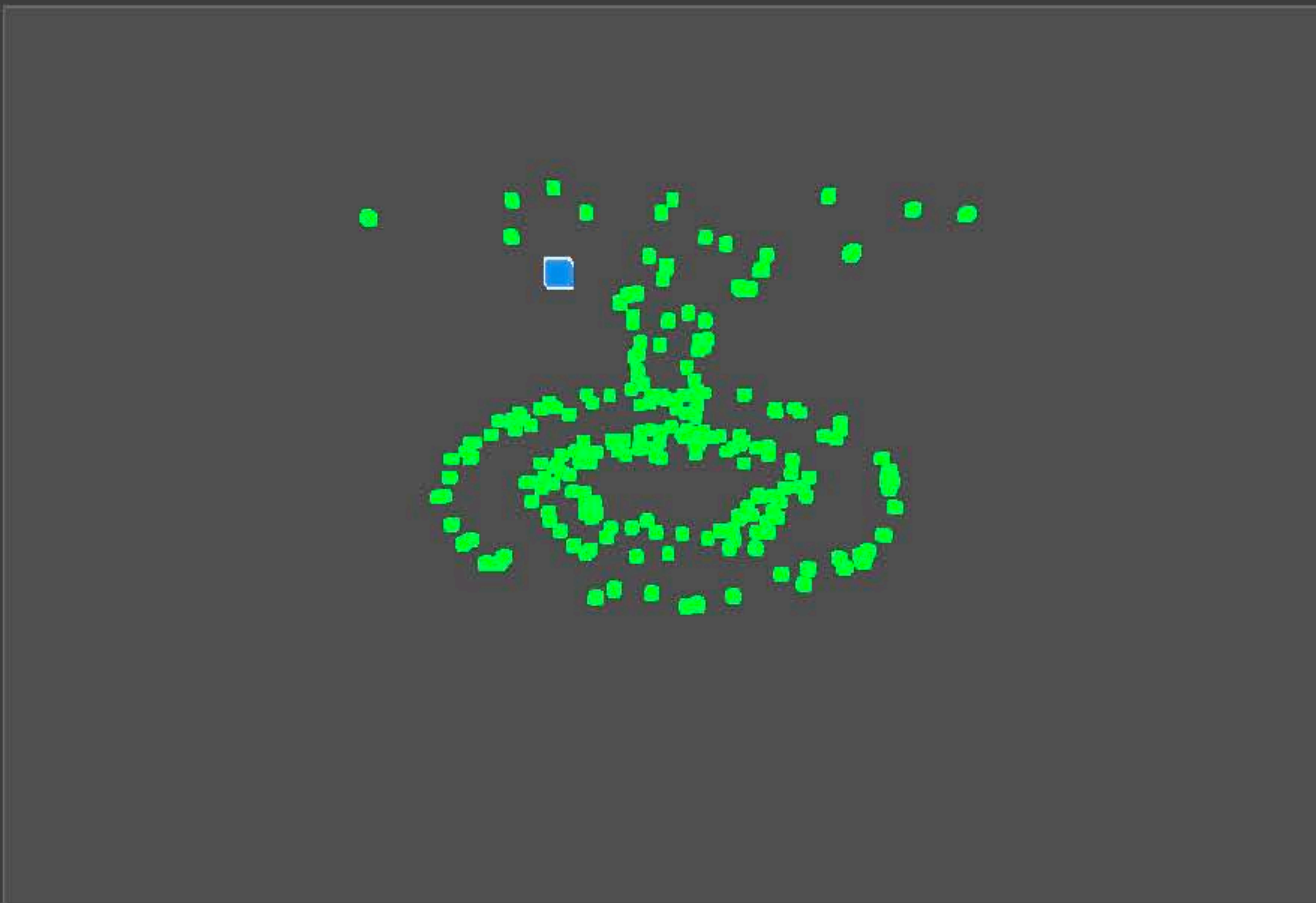
8 KB Offset: 0x0

4 KB Offset: 0x0

Library 0x600001753640...

Vertex

Fragment



/Depth/Stencil

ure 0x100b18e30

View Depth Stencil" (0x100a315d0)

View Depth Stencil" (0x100a315d0)

x 2010, Depth32Float_Stencil8

Vertex	out - float4 position				out - half3 color		
0	-16.20	40.07	98.66	98.76	0.850	2.322	9.789
1	-15.87	39.84	98.66	98.76	0.850	2.322	9.789

Perspective

zNear

0.1



zFar

100.0



Field of View

65.0



Debug

	X	Y	Z
Thread Position in Grid	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>
Threads per Grid	480	368	1
<hr/>			
Thread Position in Threadgroup	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>
Threads per Threadgroup	16	16	1
Threadgroup Position in Grid	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>
Threadgroups per Grid	30	23	1
			Debug

	X	Y	Z
Thread Position in Grid	0	0	0
Threads per Grid	480	368	1
<hr/>			
Thread Position in Threadgroup	0	0	0
Threads per Threadgroup	16	16	1
Threadgroup Position in Grid	0	0	0
Threadgroups per Grid	30	23	1

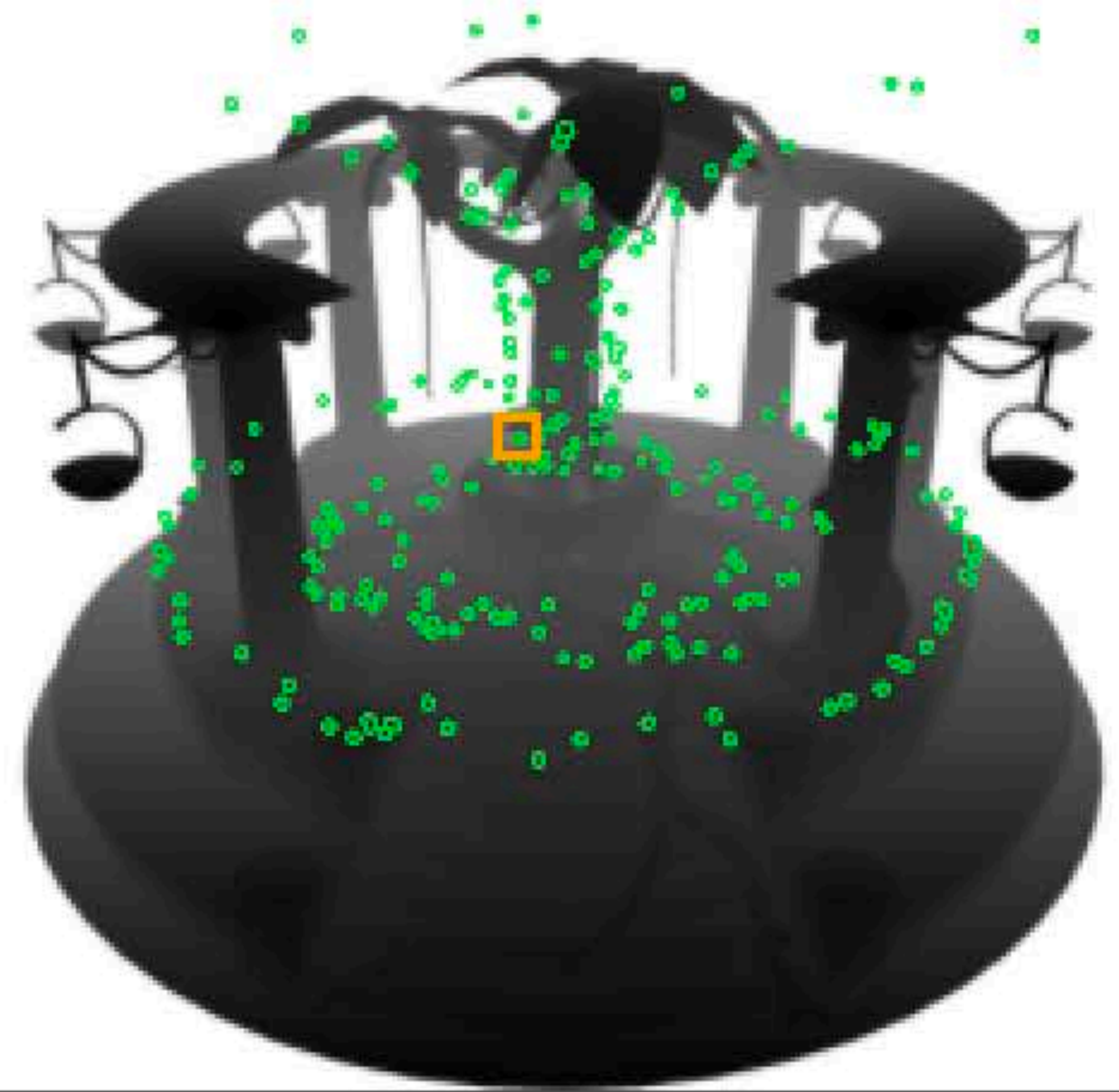
Debug

ier ,

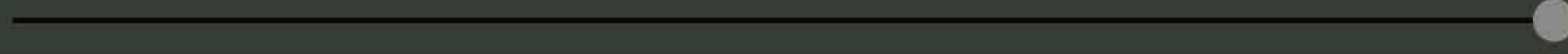
c = [0.037841797, 0.037

fragColor = [0.23693848

ret = [0.23693848, 0.37



 Depth: MTKView Depth Stencil 



[7188, 9.796875, 3.1875], [0.328125, 0.8691406]]

0]

.4]

DeferredLighting-macOS
Deferred Lighting - Debugging GPU Frame
Attachments

Deferred Lighting Captured GPU Frame

FPS: 59 FPS

GPU

X: 1,429.5
Y: 941.5
Z: 0.9991888999938965


fragment half4 fairy_fragment(FairyInOut in...)

- texture2d<half> colorMap [[texture(AAPLText...
- constexpr sampler linearSampler (mip_filter::li...
- half4 c = colorMap.sample(linearSampler, float...
- half3 fragColor = in.color * c.x;
- return half4(fragColor, c.x);

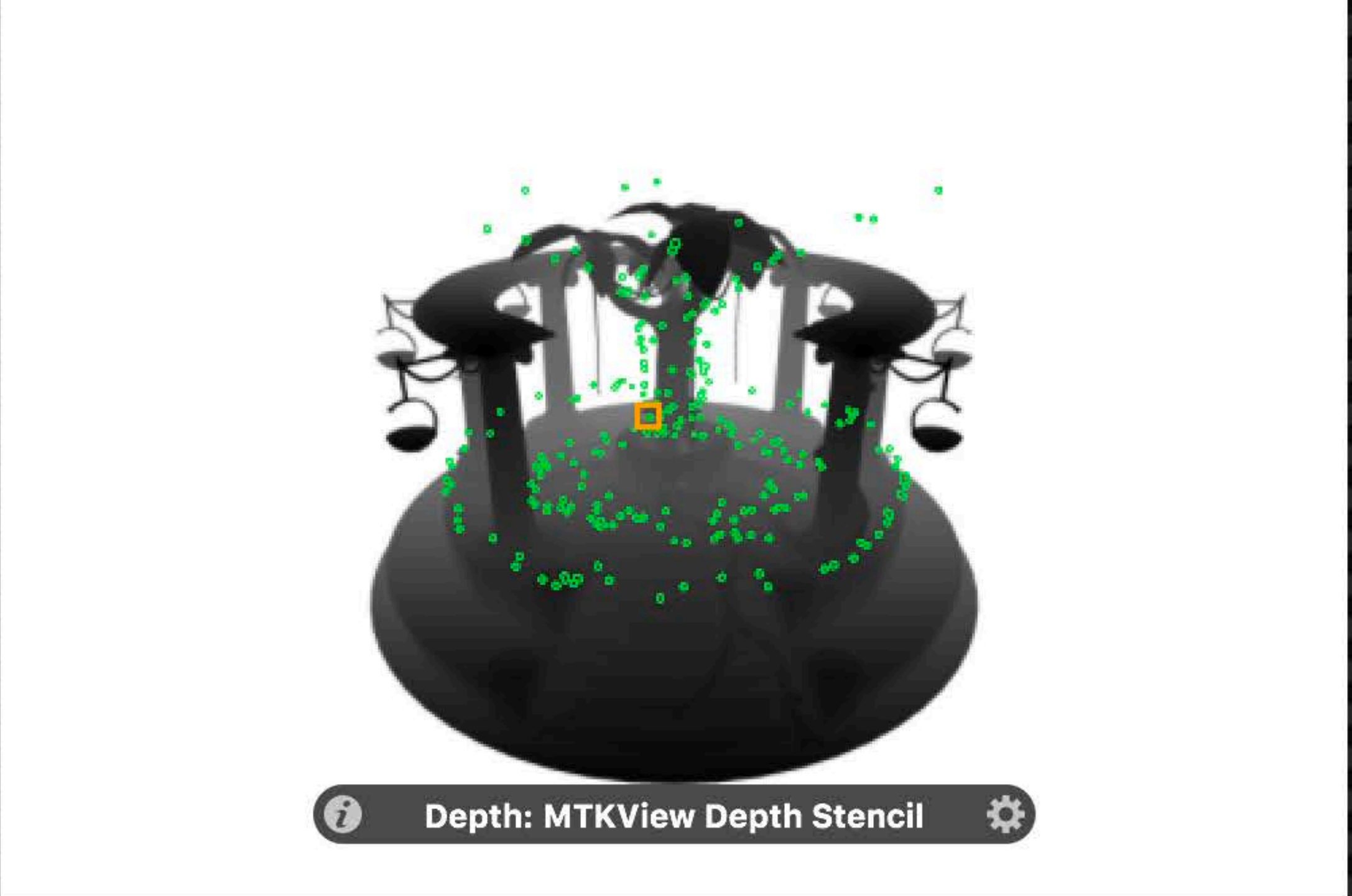
```

51
52 fragment half4 fairy_fragment(FairyInOut in in = [[1429.5, 941.5, 0.9
53     [[ stage_in ]],
54     texture2d<half> colorMap = Fairy Map
55     colorMap
56     [[ texture(AAPLT
57     extureIndexAlpha
58     ) ]])
59 {
60     constexpr sampler linearSampler
61     (mip_filter::linear,
62     mag_filter::line
63     ar,
64     min_filter::line
65     ar);
66
67     //#-code-listing(renderFairyLights)
68     half4 c = colorMap.sample(linearSampler,
69     float2(in.tex_coord));
70
71     half3 fragColor = in.color * c.x;
72     fragColor = [0.23693848
73
74     return half4(fragColor, c.x);
75     ret = [0.23693848, 0.37
76
77     //#-end-code-listing
78 }
79
80

```



Color 0



Depth: MTKView Depth Stencil

No Selection

in = (FairyInOut) [[1429.5, 941.5, 0.9991889, 0.008609643], [6.2617188, 9.796875, 3.1875], [0.328125, 0.8691406]]

colorMap = (texture2d<half, metal::access::sample, void>) [0x10080fcb0]

linearSampler = (sampler) [0x782481bffc002034]

c = (half4) [0.037841797, 0.037841797, 0.037841797, 0.99902344]

Inspecting Variables

Just go the source line

```
mag_filter::linear);

half3 tangent_normal =
    bump_texture.sample(linear_sampler,
        in.v_texcoord.xy).xyz * 2.0 - 1.0;
float4 albedo = albedo_texture.sample(linear_sampler,
    in.v_texcoord.xy);

half specular_mask =
    specular_texture.sample(linear_sampler,
        in.v_texcoord.xy).r;
float3 normal = calcNorm(tangent_normal, in);

float scale = rsqrt(dot(normal, normal)) * 0.5;

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

float r =
    shadow_texture.sample_compare(shadow_sampler,
        in.v_shadowcoord.xy, in.v_shadowcoord.z);

FragOutput output;
```

tangent_normal = [0.021, 0.021, 0.021]
albedo = [0.174197, 0.174197, 0.174197, 0.174197]
specular_mask = 0.0064
normal = [0.187226, 0.187226, 0.187226]
scale = 0.621507
shadow_sampler = compare_func::less
r = 0.0

Inspecting Variables

Just go the source line

- The side bar for the modified variables

```
mag_filter::linear);  
  
half3 tangent_normal =  
    bump_texture.sample(linear_sampler,  
        in.v_texcoord.xy).xyz * 2.0 - 1.0;  
float4 albedo = albedo_texture.sample(linear_sampler,  
        in.v_texcoord.xy);  
  
half specular_mask =  
    specular_texture.sample(linear_sampler,  
        in.v_texcoord.xy).r;  
float3 normal = calcNorm(tangent_normal, in);  
float scale = rsqrt(dot(normal, normal)) * 0.5;  
  
constexpr sampler shadow_sampler(coord::normalized,  
    filter::linear, address::clamp_to_edge,  
    compare_func::less);  
  
float r =  
    shadow_texture.sample_compare(shadow_sampler,  
        in.v_shadowcoord.xy, in.v_shadowcoord.z);  
  
FragOutput output;
```

```
tangent_normal = [0.021 0.021 0.021]   
albedo = [0.174197, 0.174197, 0.174197]   
specular_mask = 0.0064   
normal = [0.187226, 0.187226, 0.187226]   
scale = 0.621507   
shadow_sampler = com.   
r = 0.0   
  

```


Inspecting Variables

Just go the source line

- The side bar for the modified variables

```
mag_filter::linear);

half3 tangent_normal =
    bump_texture.sample(linear_sampler,
        in.v_texcoord.xy).xyz * 2.0 - 1.0;
float4 albedo = albedo_texture.sample(linear_sampler,
    in.v_texcoord.xy);

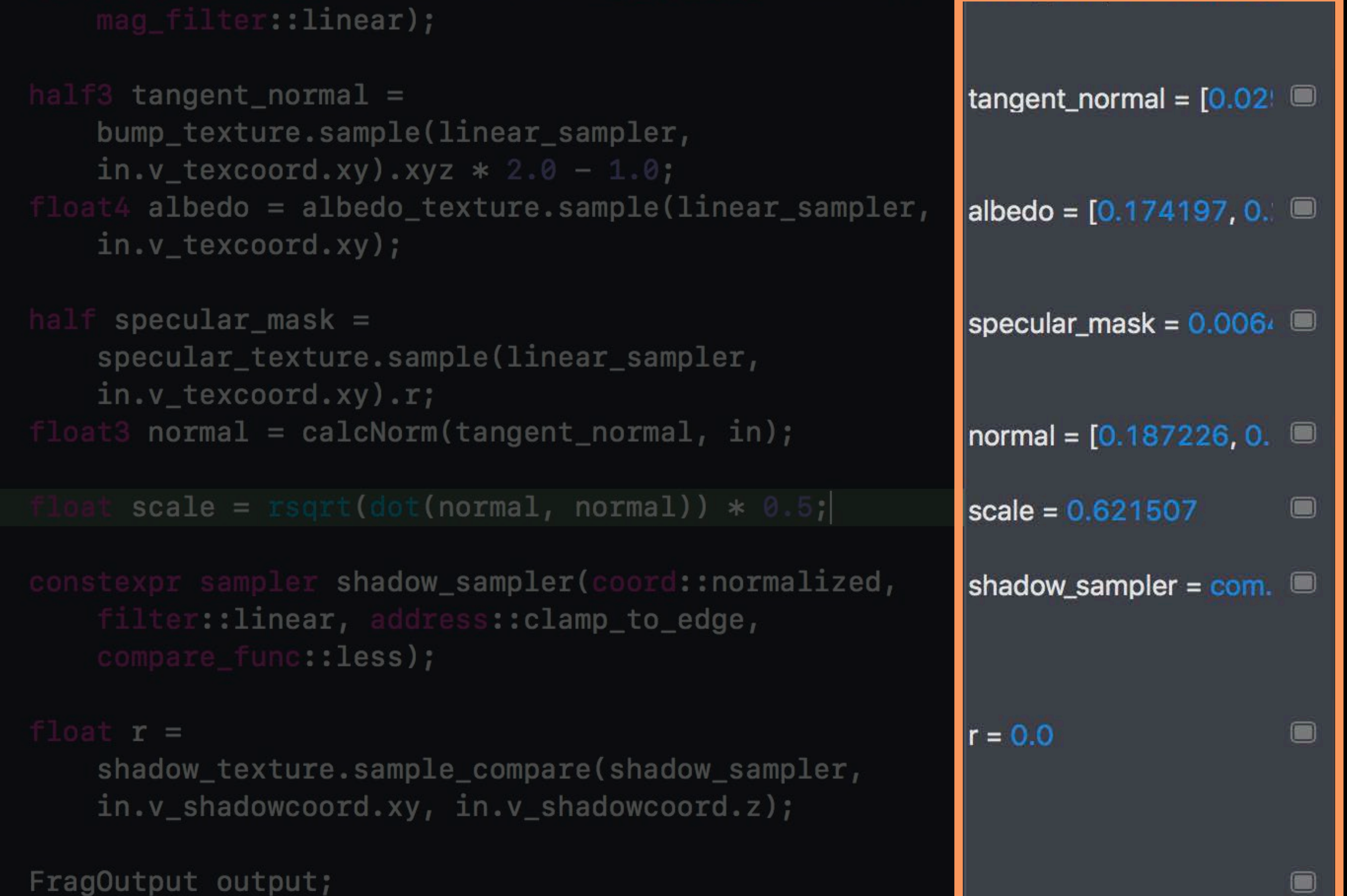
half specular_mask =
    specular_texture.sample(linear_sampler,
        in.v_texcoord.xy).r;
float3 normal = calcNorm(tangent_normal, in);

float scale = rsqrt(dot(normal, normal)) * 0.5;

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

float r =
    shadow_texture.sample_compare(shadow_sampler,
        in.v_shadowcoord.xy, in.v_shadowcoord.z);

FragOutput output;
```



tangent_normal = [0.02, 0.02, 0.02]
albedo = [0.174197, 0.174197, 0.174197, 0.174197]
specular_mask = 0.0064
normal = [0.187226, 0.187226, 0.187226]
scale = 0.621507
shadow_sampler = com.
r = 0.0

Inspecting Variables

Just go the source line

- The side bar for the modified variables
- Details view for full information

```
FragOutput output;

if(shouldApplyScale(in))
{
    normal = normal * scale + 0.5;
}

output.albedo.rgb = albedo.rgb;
output.albedo.a = r;
output.normal.rgb = normal.xyz;
output.normal.w = specular_mask;
output.depth = in.v_lineardepth;
output.light = clear_color_gbuffer3;
return output;
```

output = (FragOutput) [[0.174197, 0.213498, 0.0666904, 0.0], [0.187226, 0.158399, -0.766205, 0.00649261], 9.76893, [0.1, 0.1, 0.125, 0.0]]

- ▶ albedo = (float4) [0.174197, 0.213498, 0.0666904, 0.0]
- ▶ normal = (float4) [0.187226, 0.158399, -0.766205, 0.00649261]
- ▶ depth = (float) 9.76893
- ▶ light = (float4) [0.1, 0.1, 0.125, 0.0]

ret = [[0.174197, 0.213498, 0.0666904, 0.0], [0.187226, 0.158399, -0.766205, 0.00649261], 9.76893, [0.1, 0.1, 0.125, 0.0]]

Inspecting Variables

Just go the source line

- The side bar for the modified variables
- Details view for full information

```
FragOutput output;

if(shouldApplyScale(in))
{
    normal = normal * scale + 0.5;
}

output.albedo.rgb = albedo.rgb;
output.albedo.a = r;
output.normal.rgb = normal.xyz;
output.normal.w = specular_mask;

output.light = clear_color_gbuffer3;

return output;
```

output = [[0.174197, 0.213498, 0.0666904, 0.0], [0.187226, 0.158399, -0.766205, 0.00649261], 9.76893, [0.1, 0.1, 0.125, 0.0]]

output = [[0.174197, 0.213498, 0.0666904, 0.0], [0.187226, 0.158399, -0.766205, 0.00649261], 9.76893, [0.1, 0.1, 0.125, 0.0]]

ret = [[0.174197, 0.213498, 0.0666904, 0.0], [0.187226, 0.158399, -0.766205, 0.00649261], 9.76893, [0.1, 0.1, 0.125, 0.0]]

Inspecting Variables

Just go the source line

- The side bar for the modified variables
- Details view for full information

```
FragOutput output;

if(shouldApplyScale(in))
{
    normal = normal * scale + 0.5;
}

output.albedo.rgb = albedo.rgb;
output.albedo.a = r;
output.normal.rgb = normal.xyz;
output.normal.w = specular_mask;
output.depth = in.v.lineardepth;
output.light = clear_color_gbuffer3;
return output;
```

output = [[0.174197, 0.213498, 0.0666904, 0.0], [0.187226, 0.158399, -0.766205, 0.00649261], 9.76893, [0.1, 0.1, 0.125, 0.0]]

ret = [[0.174197, 0.213498, 0.0666904, 0.0], [0.187226, 0.158399, -0.766205, 0.00649261], 9.76893, [0.1, 0.1, 0.125, 0.0]]

Inspecting Variables

Just go the source line

- The side bar for the modified variables
- Details view for full information
- Hover for in place access

```
shadowMap
[[ texture(AAPLTextu
reIndexShadow) ]])

constexpr sampler linearSampler(mip_filter::linear,
mag_filter::linear,
min_filter::linear);

half4 base_color_sample =
baseColorMap.sample(linearSampler, in.tex_coord.xy);
half4 normal_sample = normalMap.sample(linearSampler,
baseColorMap = (texture2d<half, metal::access::sa
Pixel Format RGBA8Unorm_sRGB
Width 1,024
Height 1,024
Depth 1
sample.xyz *
2.0) - 1.0);

half3 eye_normal = (tangent_normal.x * in.tangent +
tangent_normal.y * in.bitangent +
tangent_normal.z * in.normal);
```

shadowMap = [0x1718]

linearSampler = [0x782]

base_color_sample = [0]

normal_sample = [0.614]

specular_contrib = 0.09

gBuffer = [[n/a, n/a, n/a, n/a]

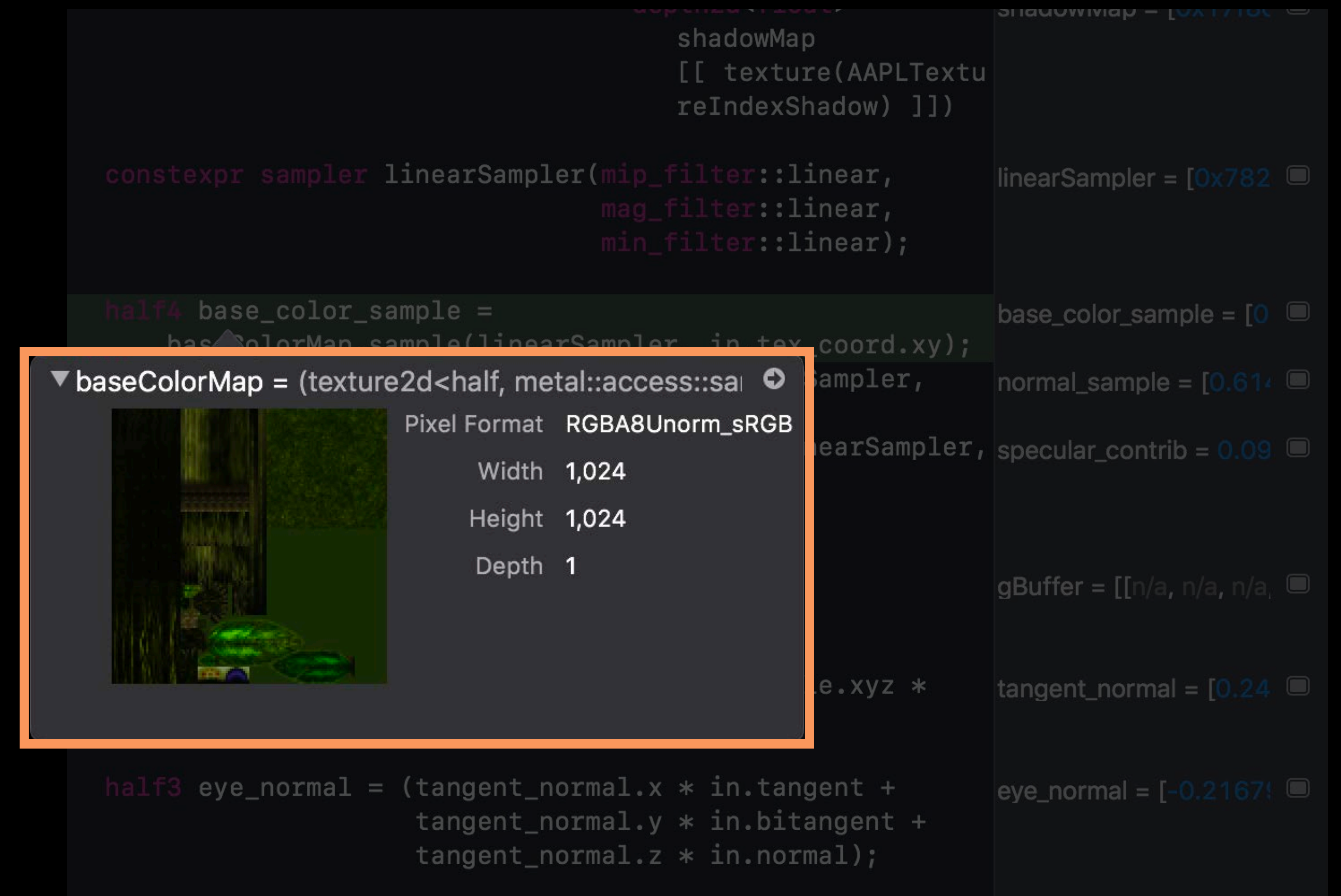
tangent_normal = [0.24]

eye_normal = [-0.2167]

Inspecting Variables

Just go the source line

- The side bar for the modified variables
- Details view for full information
- Hover for in place access



Inspecting Variables

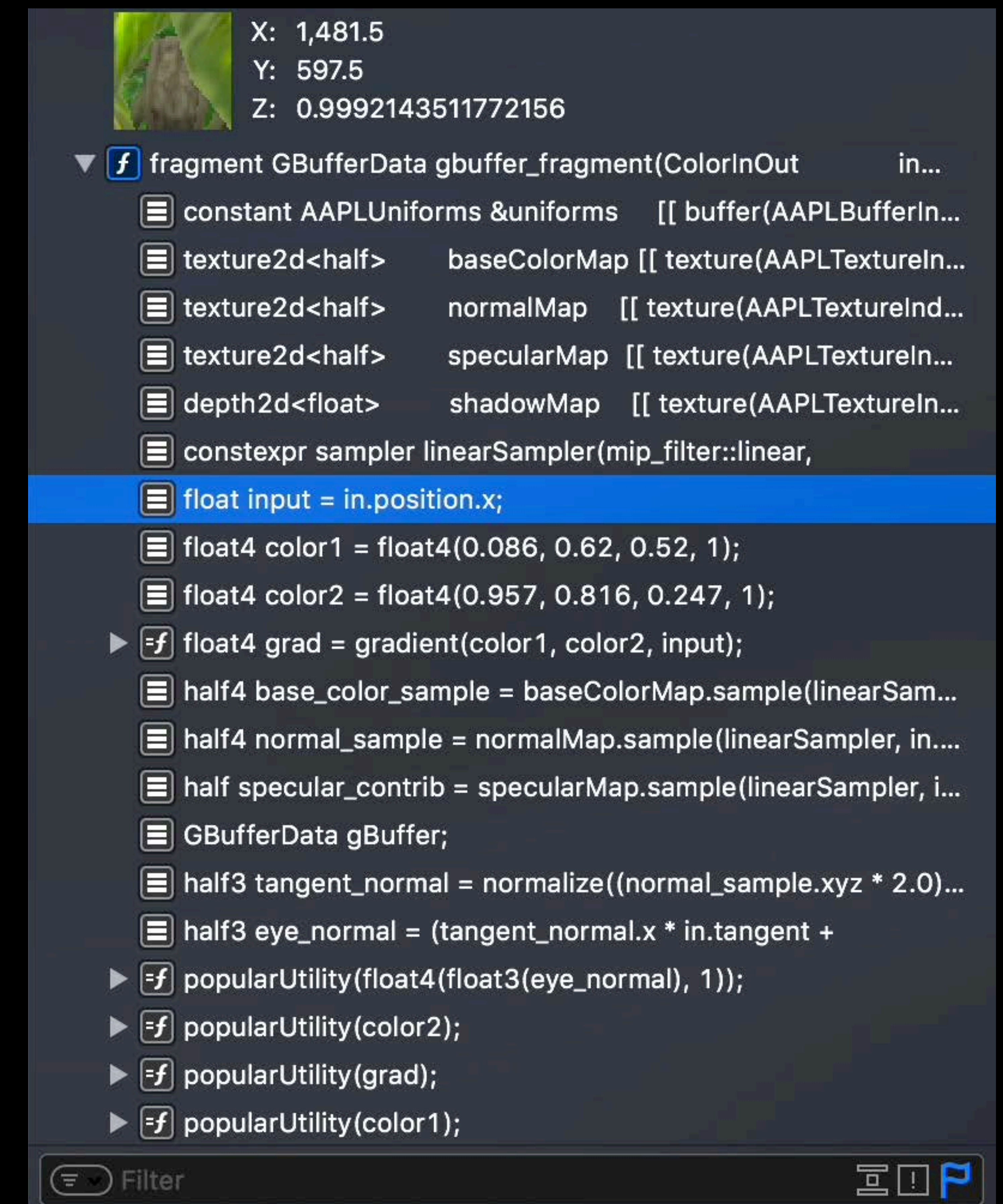
Just go the source line

- The side bar for the modified variables
- Details view for full information
- Hover for in place access
- Variables view for variables in scope

```
▶ in = (VertexOutput) [[524.5, 379.5, 0.993738, 0.102365], [0.456372, 0.471244, 0.3  
▶ clear_color_gbuffer3 = (float4) [0.1, 0.1, 0.125, 0.0]  
▶ bump_texture = (texture2d<half, metal::access::sample, void>) [0x101058200]  
▶ albedo_texture = (texture2d<float, metal::access::sample, void>) [0x10188fa00]  
▶ specular_texture = (texture2d<half, metal::access::sample, void>) [0x1020a7800]  
▶ shadow_texture = (depth2d<float, metal::access::sample, void>) [0x102044400]  
▶ linear_sampler = (sampler) [0x600002601da0]  
▶ tangent_normal = (half3) [0.0292969, -0.0981445, 0.985352]  
▶ albedo = (float4) [0.174197, 0.213498, 0.0666904, 1.0]  
  specular_mask = (half) 0.00649261  
▶ normal = (float3) [0.187226, 0.158399, -0.766205]  
  scale = (float) 0.621507
```


Following the Execution

All source lines executed are available in navigator



X: 1,481.5
Y: 597.5
Z: 0.9992143511772156

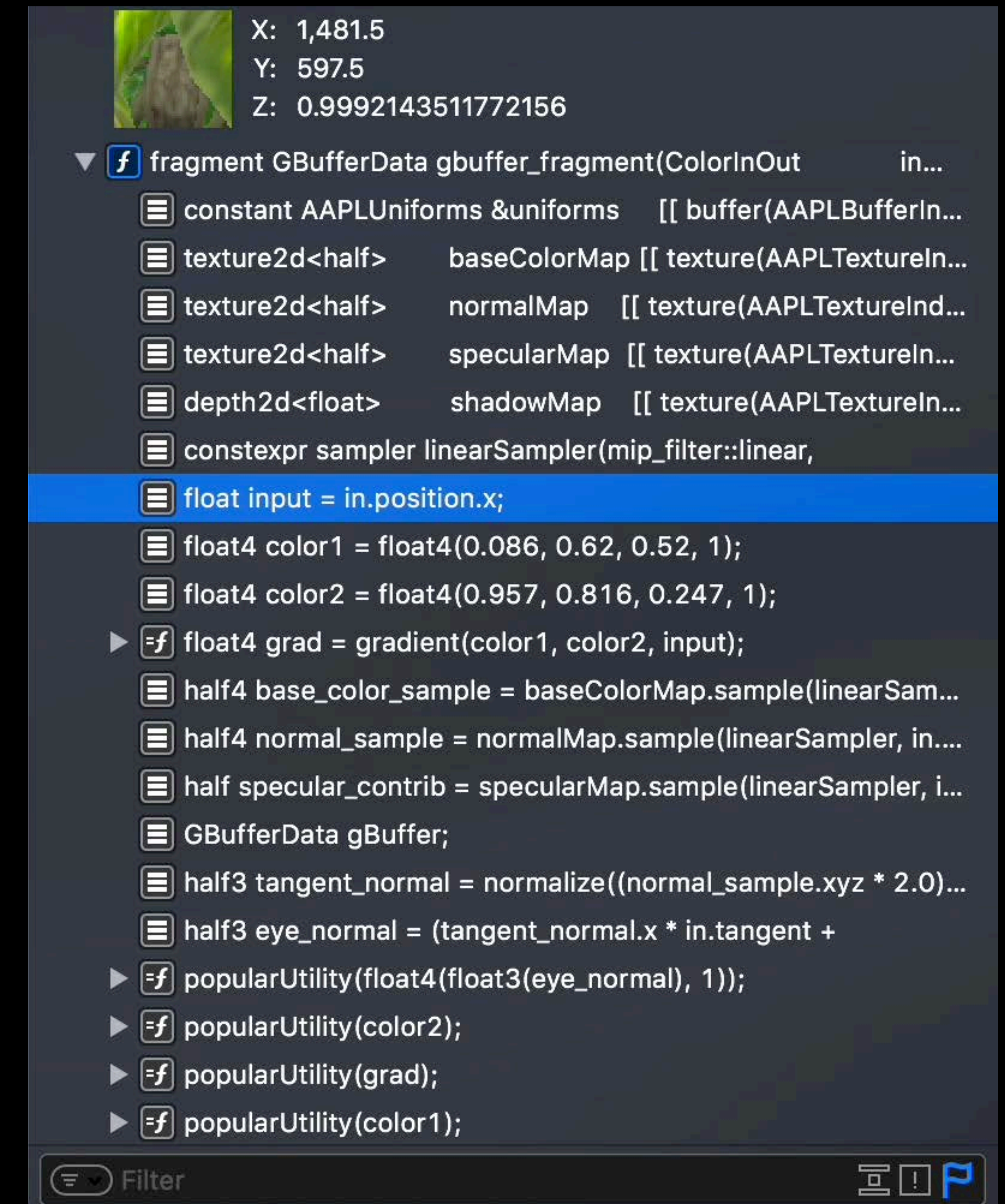
```
fragment GBufferData gbuffer_fragment(ColorInOut in...
  constant AAPLUniforms &uniforms  [[ buffer(AAPLBufferIn...
  texture2d<half>    baseColorMap  [[ texture(AAPLTextureIn...
  texture2d<half>    normalMap    [[ texture(AAPLTextureInd...
  texture2d<half>    specularMap  [[ texture(AAPLTextureIn...
  depth2d<float>    shadowMap    [[ texture(AAPLTextureIn...
  constexpr sampler linearSampler(mip_filter::linear,
  float input = in.position.x;
  float4 color1 = float4(0.086, 0.62, 0.52, 1);
  float4 color2 = float4(0.957, 0.816, 0.247, 1);
  float4 grad = gradient(color1, color2, input);
  half4 base_color_sample = baseColorMap.sample(linearSam...
  half4 normal_sample = normalMap.sample(linearSampler, in...
  half specular_contrib = specularMap.sample(linearSampler, i...
  GBufferData gBuffer;
  half3 tangent_normal = normalize((normal_sample.xyz * 2.0)...
  half3 eye_normal = (tangent_normal.x * in.tangent +
  popularUtility(float4(float3(eye_normal), 1));
  popularUtility(color2);
  popularUtility(grad);
  popularUtility(color1);
```

Filter

Following the Execution

All source lines executed are available in navigator

- Flexible stepping

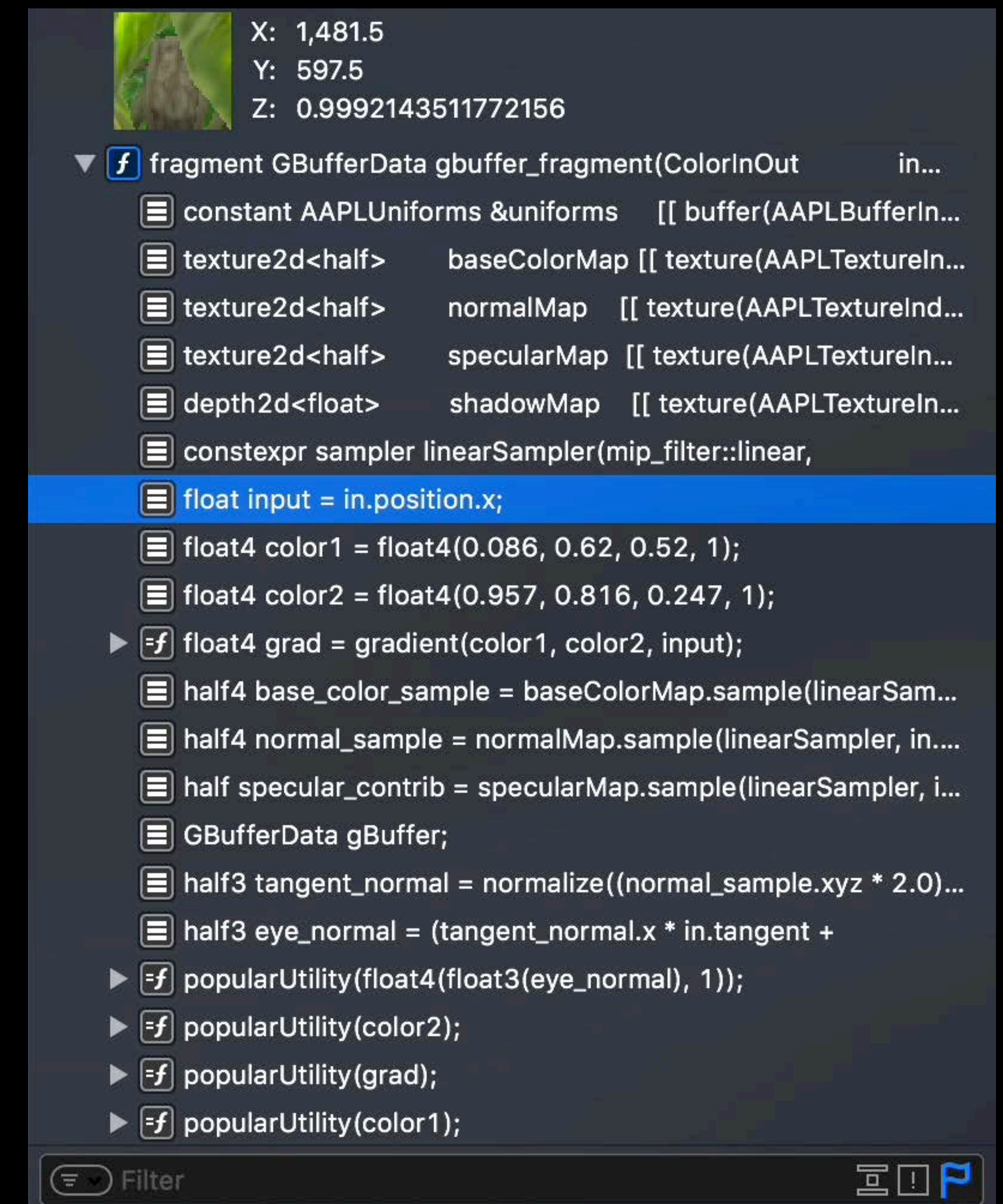


```
X: 1,481.5  
Y: 597.5  
Z: 0.9992143511772156  
▼ f fragment GBufferData gbuffer_fragment(ColorInOut in...  
  ▢ constant AAPLUniforms &uniforms [[ buffer(AAPLBufferIn...  
  ▢ texture2d<half> baseColorMap [[ texture(AAPLTextureIn...  
  ▢ texture2d<half> normalMap [[ texture(AAPLTextureInd...  
  ▢ texture2d<half> specularMap [[ texture(AAPLTextureIn...  
  ▢ depth2d<float> shadowMap [[ texture(AAPLTextureIn...  
  ▢ constexpr sampler linearSampler(mip_filter::linear,  
  ▢ float input = in.position.x;  
  ▢ float4 color1 = float4(0.086, 0.62, 0.52, 1);  
  ▢ float4 color2 = float4(0.957, 0.816, 0.247, 1);  
  ▶ ▢ float4 grad = gradient(color1, color2, input);  
  ▢ half4 base_color_sample = baseColorMap.sample(linearSam...  
  ▢ half4 normal_sample = normalMap.sample(linearSampler, in...  
  ▢ half specular_contrib = specularMap.sample(linearSampler, i...  
  ▢ GBufferData gBuffer;  
  ▢ half3 tangent_normal = normalize((normal_sample.xyz * 2.0)...  
  ▢ half3 eye_normal = (tangent_normal.x * in.tangent +  
  ▶ ▢ popularUtility(float4(float3(eye_normal), 1));  
  ▶ ▢ popularUtility(color2);  
  ▶ ▢ popularUtility(grad);  
  ▶ ▢ popularUtility(color1);
```


Following the Execution

All source lines executed are available in navigator

- Flexible stepping
 - Backwards debugging



```
X: 1,481.5
Y: 597.5
Z: 0.9992143511772156

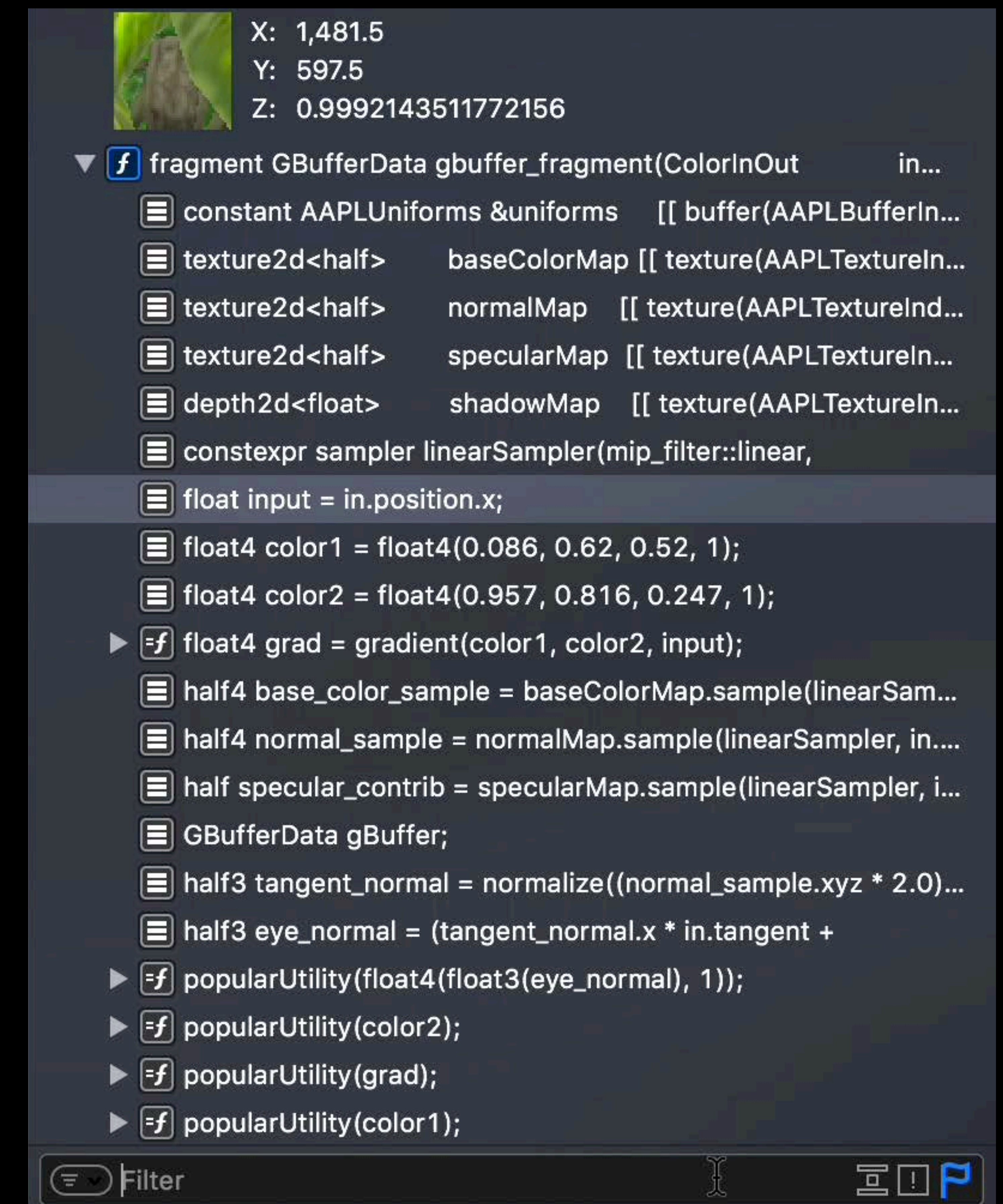
▼ f fragment GBufferData gbuffer_fragment(ColorInOut in...
  ▣ constant AAPLUniforms &uniforms [[ buffer(AAPLBufferIn...
  ▣ texture2d<half> baseColorMap [[ texture(AAPLTextureIn...
  ▣ texture2d<half> normalMap [[ texture(AAPLTextureInd...
  ▣ texture2d<half> specularMap [[ texture(AAPLTextureIn...
  ▣ depth2d<float> shadowMap [[ texture(AAPLTextureIn...
  ▣ constexpr sampler linearSampler(mip_filter::linear,
  ▣ float input = in.position.x;
  ▣ float4 color1 = float4(0.086, 0.62, 0.52, 1);
  ▣ float4 color2 = float4(0.957, 0.816, 0.247, 1);
  ▶ ▣ float4 grad = gradient(color1, color2, input);
  ▣ half4 base_color_sample = baseColorMap.sample(linearSam...
  ▣ half4 normal_sample = normalMap.sample(linearSampler, in...
  ▣ half specular_contrib = specularMap.sample(linearSampler, i...
  ▣ GBufferData gBuffer;
  ▣ half3 tangent_normal = normalize((normal_sample.xyz * 2.0)...
  ▣ half3 eye_normal = (tangent_normal.x * in.tangent +
  ▶ ▣ popularUtility(float4(float3(eye_normal), 1));
  ▶ ▣ popularUtility(color2);
  ▶ ▣ popularUtility(grad);
  ▶ ▣ popularUtility(color1);
```

Filter

Following the Execution

All source lines executed are available in navigator

- Flexible stepping
 - Backwards debugging
- Use filter to focus



X: 1,481.5
Y: 597.5
Z: 0.9992143511772156

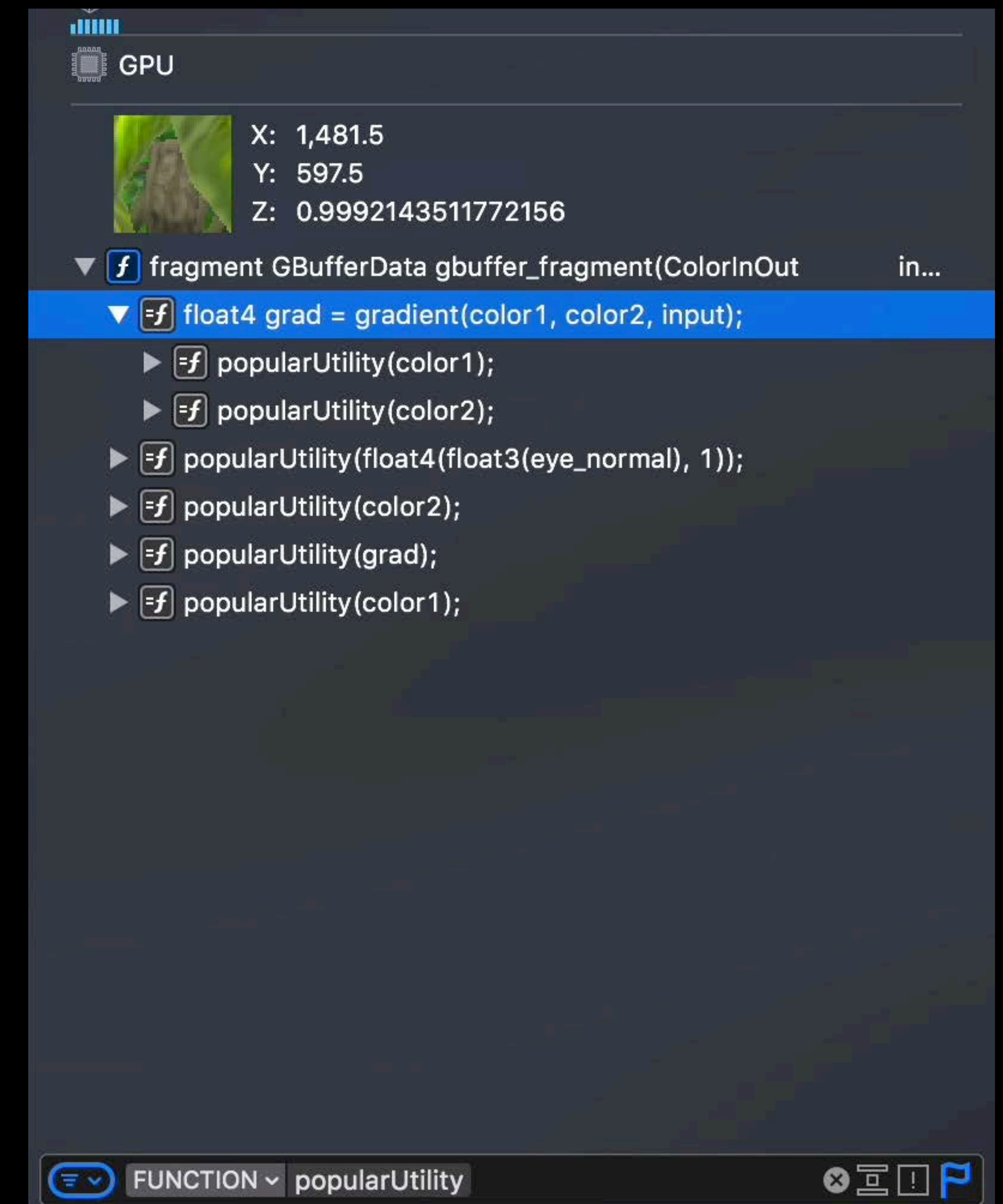
```
fragment GBufferData gbuffer_fragment(ColorInOut in...
  constant AAPLUniforms &uniforms  [[ buffer(AAPLBufferIn...
  texture2d<half>    baseColorMap  [[ texture(AAPLTextureIn...
  texture2d<half>    normalMap    [[ texture(AAPLTextureInd...
  texture2d<half>    specularMap  [[ texture(AAPLTextureIn...
  depth2d<float>    shadowMap    [[ texture(AAPLTextureIn...
  constexpr sampler linearSampler(mip_filter::linear,
  float input = in.position.x;
  float4 color1 = float4(0.086, 0.62, 0.52, 1);
  float4 color2 = float4(0.957, 0.816, 0.247, 1);
  float4 grad = gradient(color1, color2, input);
  half4 base_color_sample = baseColorMap.sample(linearSam...
  half4 normal_sample = normalMap.sample(linearSampler, in...
  half specular_contrib = specularMap.sample(linearSampler, i...
  GBufferData gBuffer;
  half3 tangent_normal = normalize((normal_sample.xyz * 2.0)...
  half3 eye_normal = (tangent_normal.x * in.tangent +
  popularUtility(float4(float3(eye_normal), 1));
  popularUtility(color2);
  popularUtility(grad);
  popularUtility(color1);
```

Filter

Following the Execution

All source lines executed are available in navigator

- Flexible stepping
 - Backwards debugging
- Use filter to focus



Access to Other Threads

Set of threads are available based on the initial selection

Access to Other Threads

Set of threads are available based on the initial selection

- Vertex — Primitive of the selected vertex

Access to Other Threads

Set of threads are available based on the initial selection

- Vertex — Primitive of the selected vertex
- Fragment — Rectangle around the selected pixel

Access to Other Threads

Set of threads are available based on the initial selection

- Vertex — Primitive of the selected vertex
- Fragment — Rectangle around the selected pixel
- Compute — Threadgroup of the selected thread

See Variables in Context

```
float4 grad = gradient(color1, color2, input);
grad = [0.76177585, 0.772069,

eye_normal = normalize(eye_normal);

constexpr sampler shadowSampler(coord::normalized,
                                filter::linear,
                                mip_filter::none,

                                address::clamp_to_e
                                dge,
                                compare_func::less);

// Compare the depth value in the shadow map to the
// depth value of the fragment in the sun's.
// frame of reference. If the sample is occluded, it
// will be zero.
//#-code-listing(occlusionTest)
float shadow_sample =
```


See Variables in Context

```
float4 grad = gradient(color1, color2, input); grad = [0.76177585, 0.772069, ]
```

```
eye_normal = normalize(eye_normal);
```

```
constexpr sampler shadowSampler(coord::normalized,  
                                filter::linear,  
                                mip_filter::none,  
  
                                address::clamp_to_e  
                                dge,  
                                compare_func::less);
```

```
// Compare the depth value in the shadow map to the  
// depth value of the fragment in the sun's.  
// frame of reference. If the sample is occluded, it  
// will be zero.
```

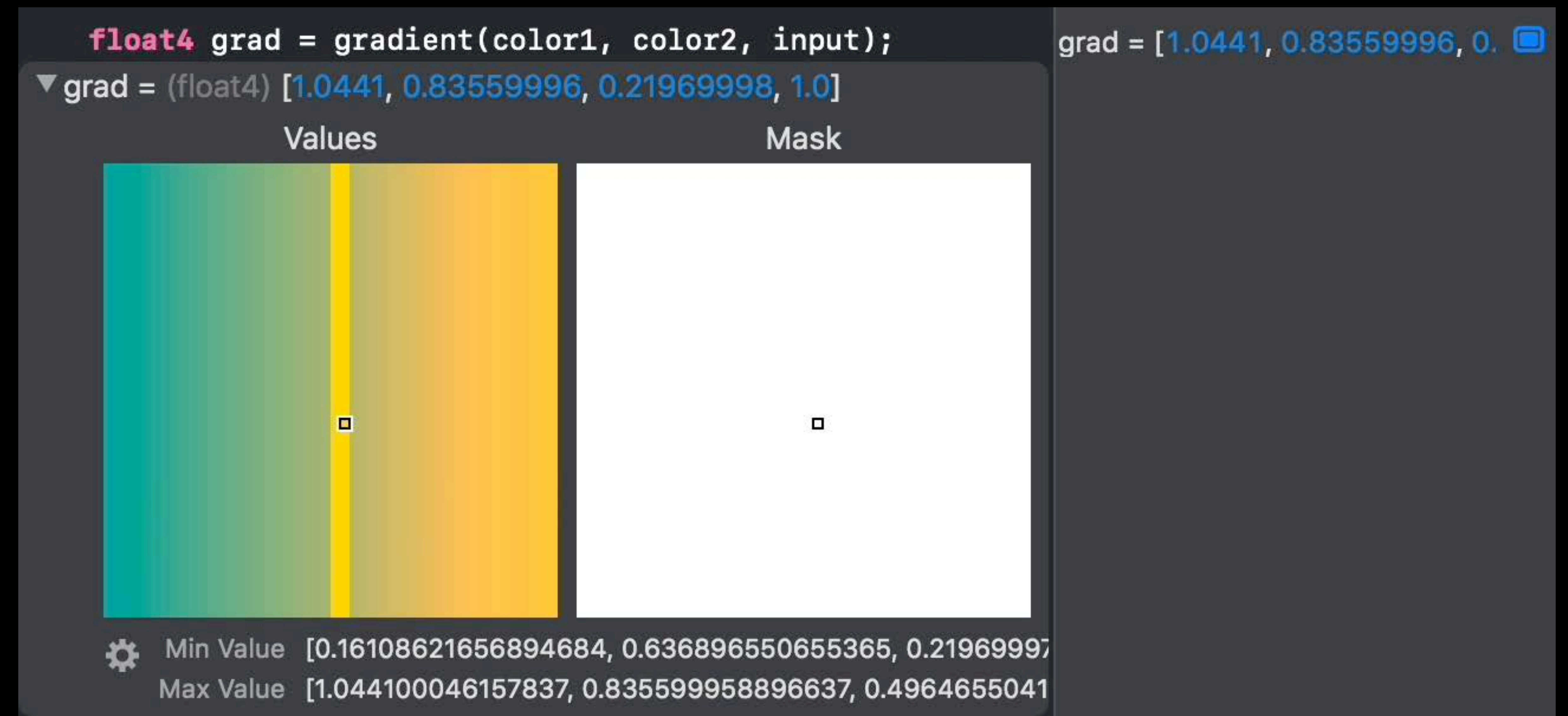
```
//#-code-listing(occlusionTest)
```

```
float shadow_sample =
```


See Variables in Context

Helps you understand your code

Important to for comparing
good/bad values

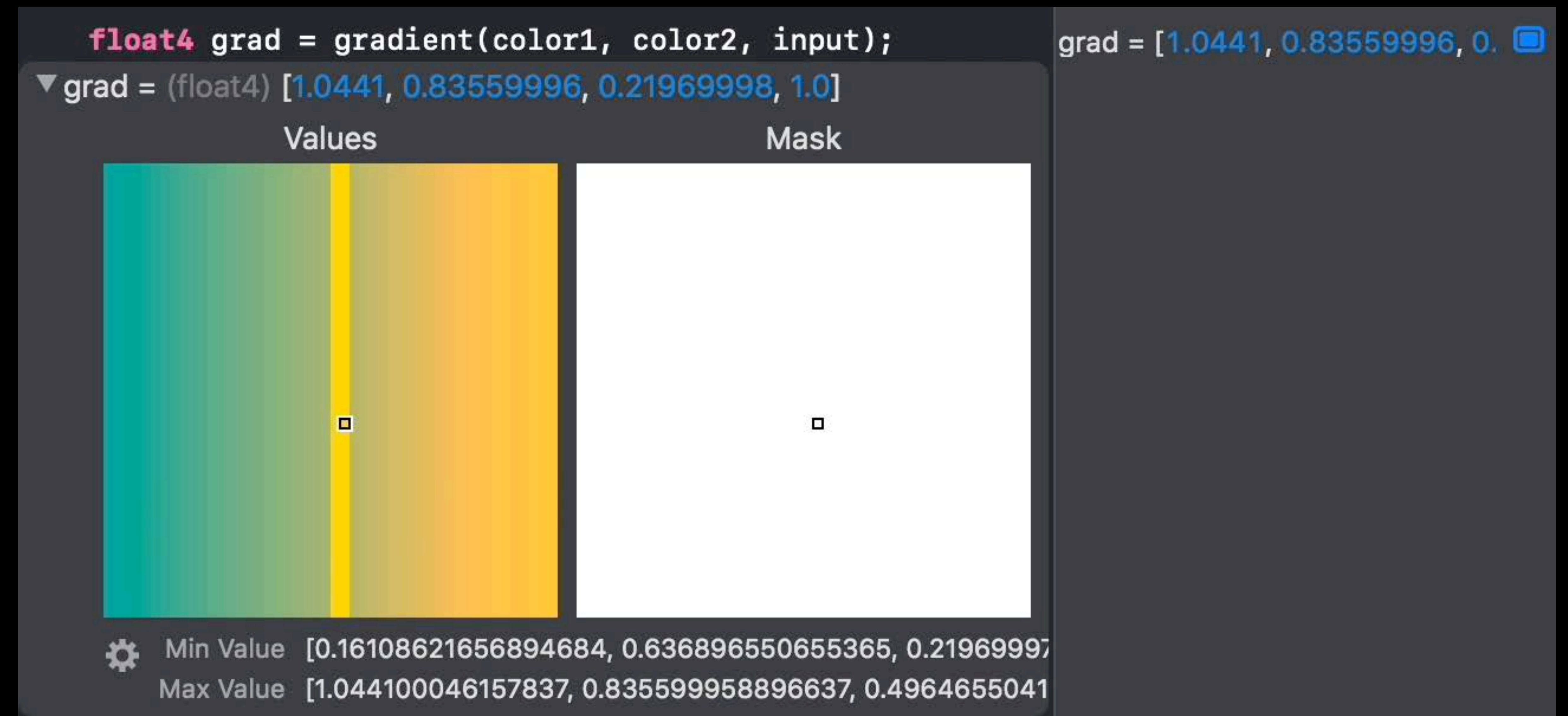


See Variables in Context

Helps you understand your code

Important to for comparing
good/bad values

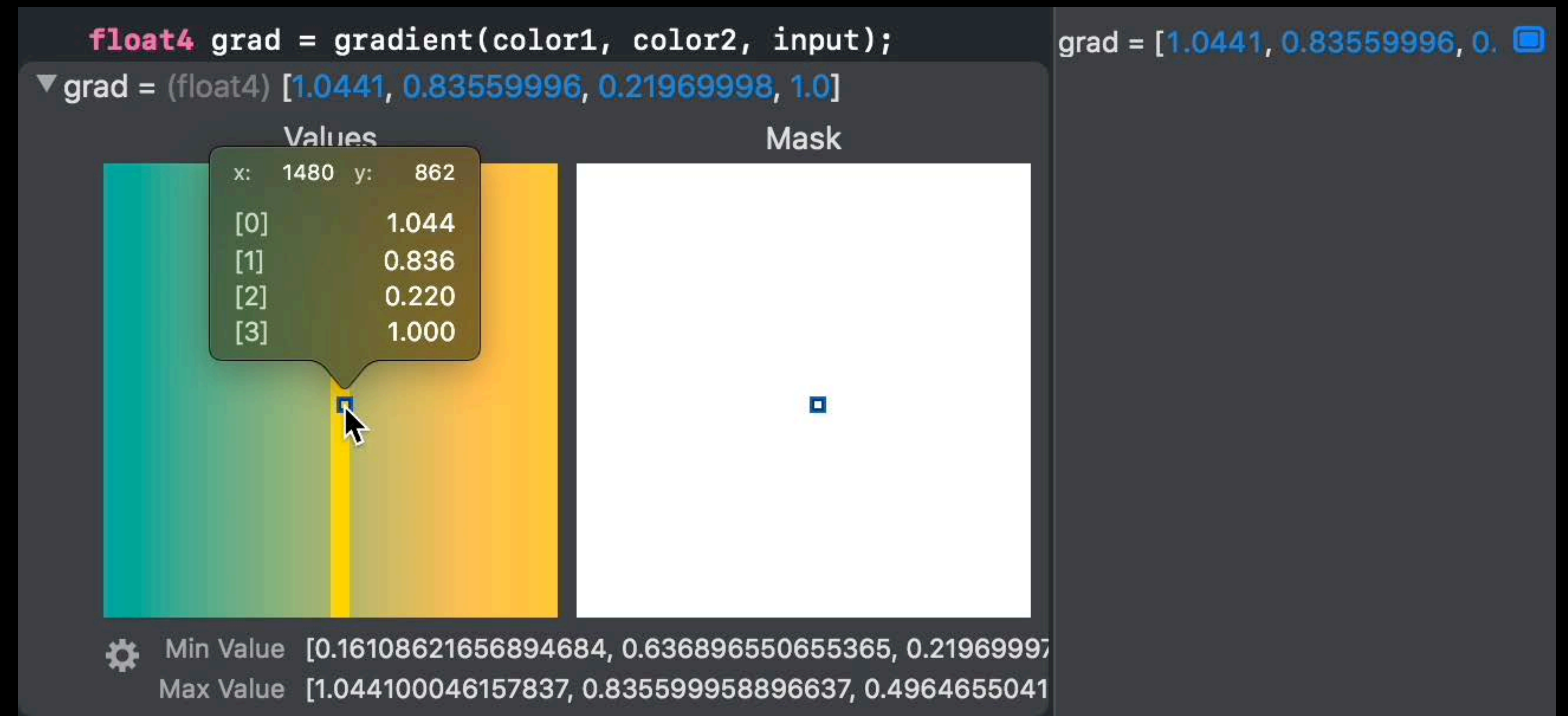
Hover for instant access



Comparing Good and Bad Pixels

Quickly compare threads

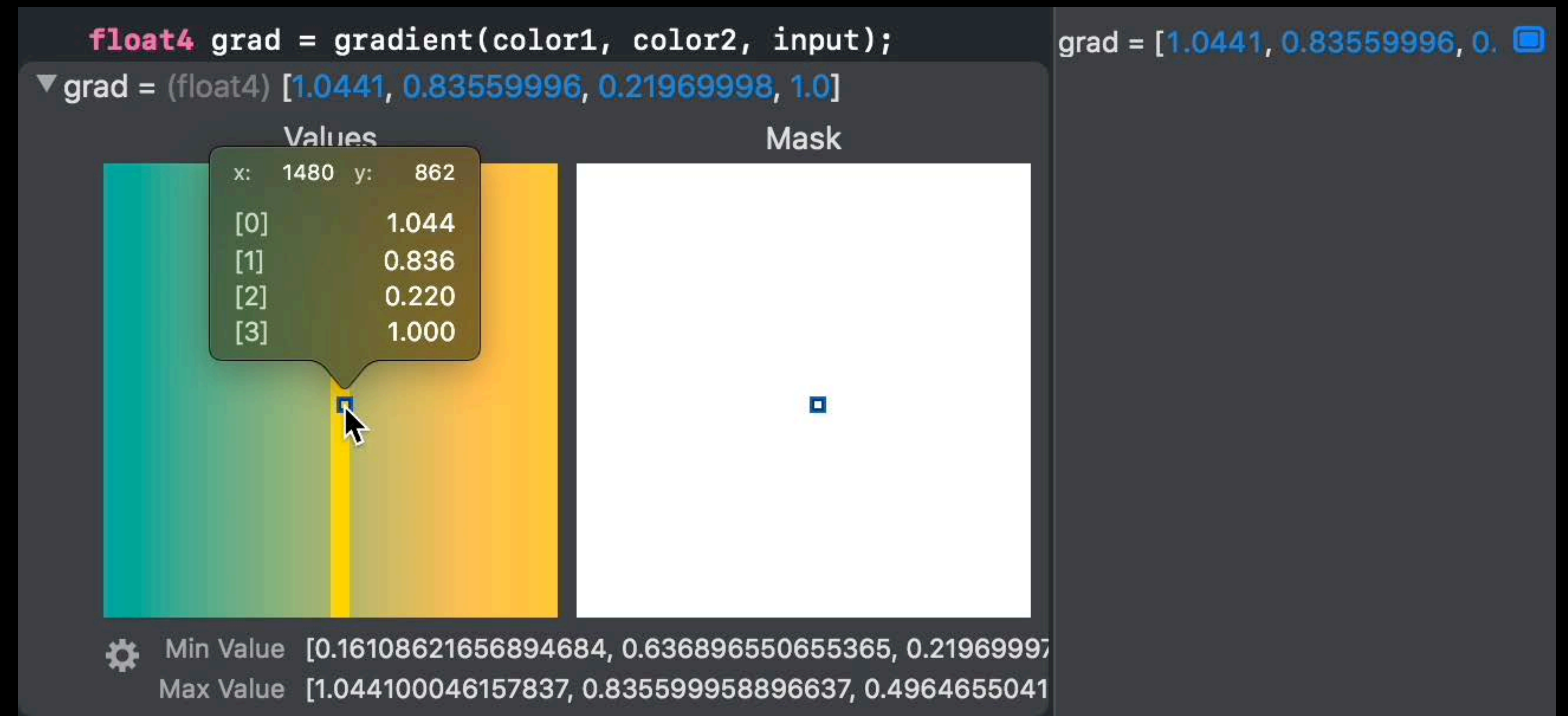
Execution history and variables are updated for the selected thread



Comparing Good and Bad Pixels

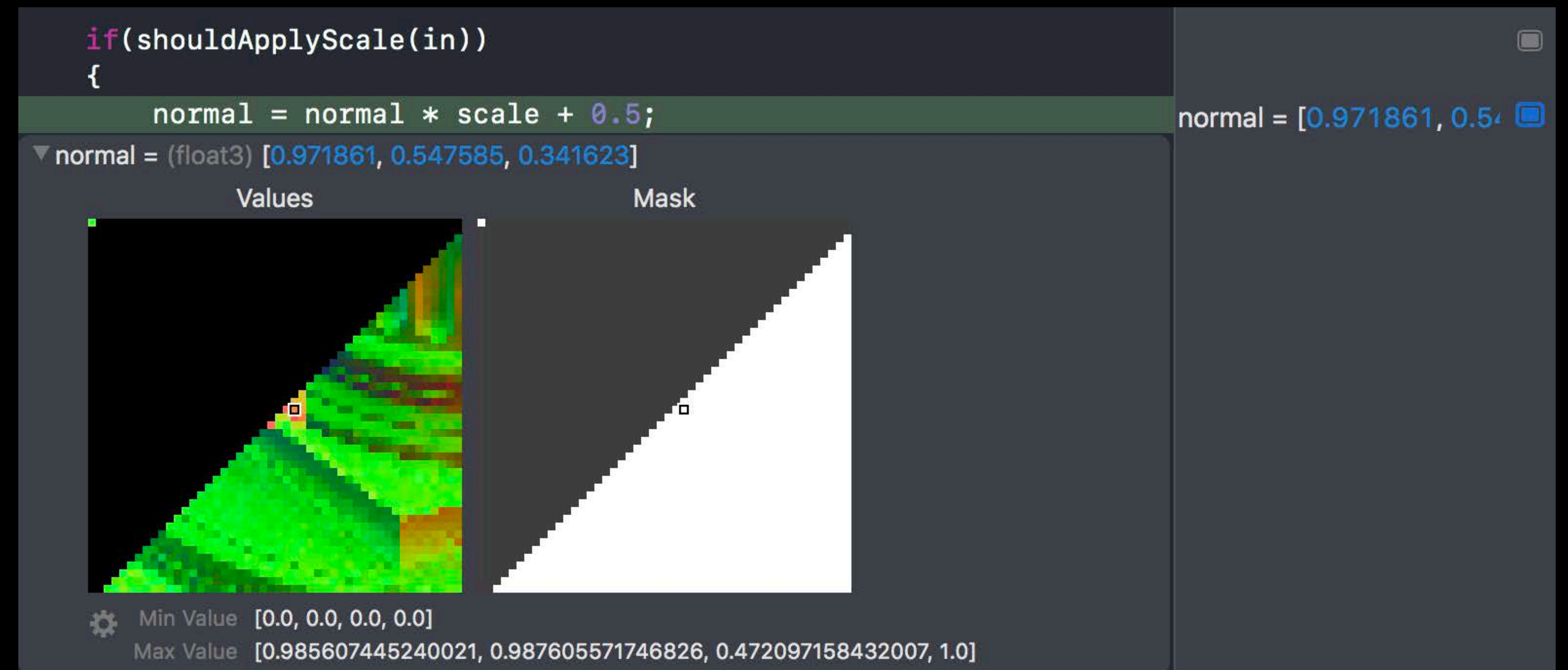
Quickly compare threads

Execution history and variables are updated for the selected thread



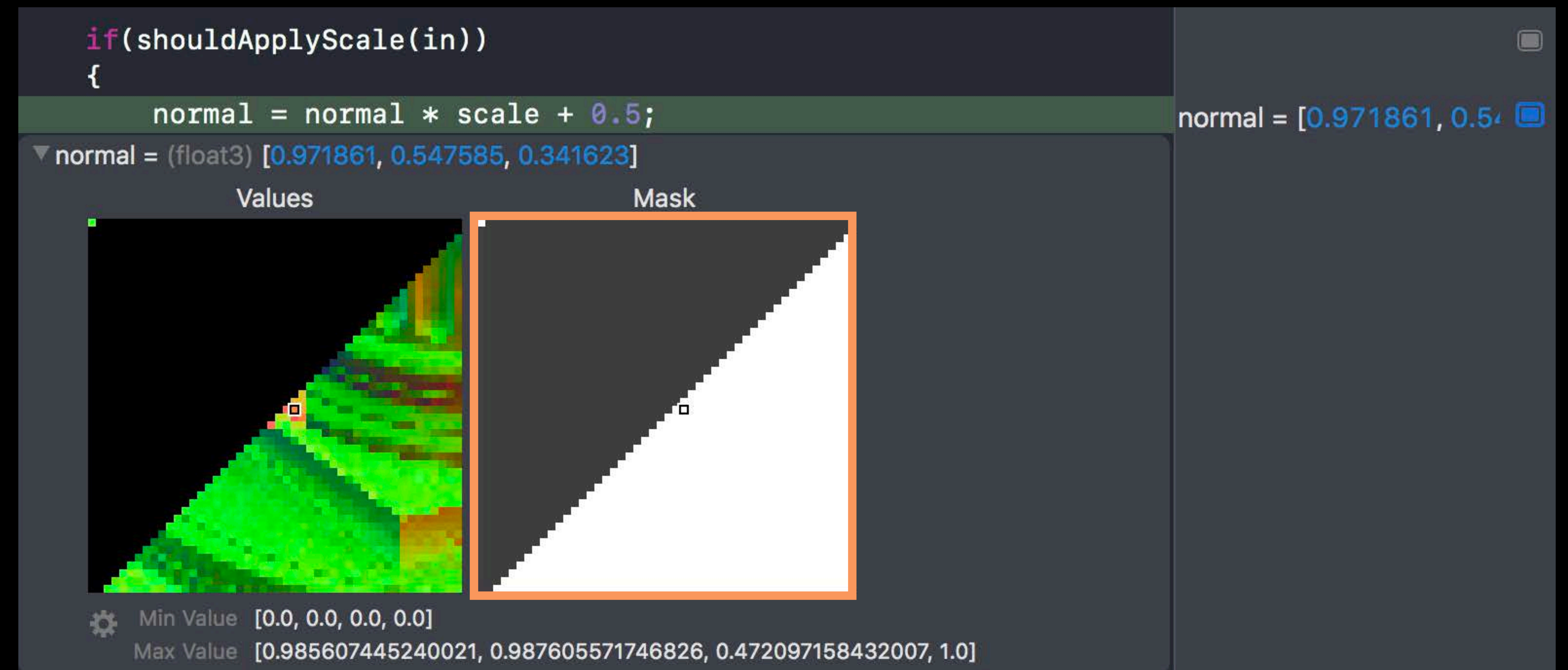
Understanding Divergence

Mask shows which threads executed the same line to help you understand control flow



Understanding Divergence

Mask shows which threads executed the same line to help you understand control flow



Demo

Xavier Verguin Gonzalez

Shader Debugger

Specifically designed for debugging Metal shaders

Shader Debugger

Specifically designed for debugging Metal shaders

Great for

- Fixing bugs!

Shader Debugger

Specifically designed for debugging Metal shaders

Great for

- Fixing bugs!
- Understanding your shader

Shader Debugger

Specifically designed for debugging Metal shaders

Great for

- Fixing bugs!
- Understanding your shader
- Developing your shaders

Shader Debugger

Specifically designed for debugging Metal shaders

Great for

- Fixing bugs!
- Understanding your shader
- Developing your shaders

Supports iOS, macOS, and tvOS

Shader Debugger

Specifically designed for debugging Metal shaders

Great for

- Fixing bugs!
- Understanding your shader
- Developing your shaders

Supports iOS, macOS, and tvOS

Available in Xcode 10

Optimizing Shaders with Shader Profiler

Knowing What to Optimize

Knowing What to Optimize

Profiling tools built into Metal Frame Debugger

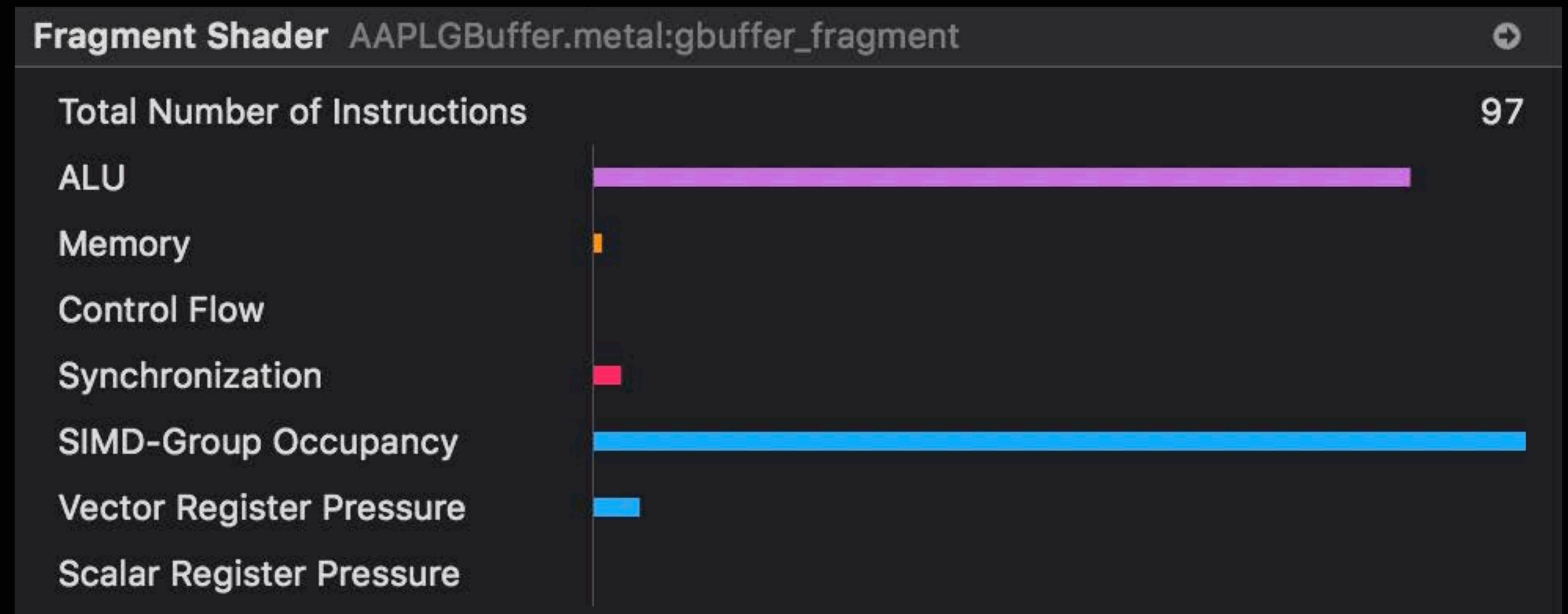
- GPU counters



Knowing What to Optimize

Profiling tools built into Metal Frame Debugger

- GPU counters
- Pipeline statistics



Knowing What to Optimize

Profiling tools built into Metal Frame Debugger

- GPU counters
- Pipeline statistics
- Shader profiler

▶ P	G-buffer Creation	1.25 ms
▶ P	Sky	661.39 μ s
▶ P	Shadow Gen	409.31 μ s
▶ P	Deferred Directional Lighting	362.65 μ s
▶ P	Light	293.16 μ s
▶ P	Point Light Mask	276.72 μ s
▶ P	Fairy Drawing	13.37 μ s

Shader Profiler

Provides per-pipeline timing information

▼	P	G-buffer Creation	1.25 ms
	S	gbuffer_vertex	1.02 ms
	S	gbuffer_fragment	1.21 ms
	▶	Folder Draws	
▶	P	Sky	661.39 μ s
▶	P	Shadow Gen	409.31 μ s
▶	P	Deferred Directional Lighting	362.65 μ s
▶	P	Light	293.16 μ s
▶	P	Point Light Mask	276.72 μ s
▶	P	Fairy Drawing	13.37 μ s

Shader Profiler

Provides per-pipeline timing information

▼ P	G-buffer Creation	1.25 ms
	S gbuffer_vertex	1.02 ms
	S gbuffer_fragment	1.21 ms
▼	Draws	
▶	42 [drawIndexedPrimitives:T...	1.01 ms
▶	46 [drawIndexedPrimitive...	225.51 μs
▶	50 [drawIndexedPrimitives:T...	7.81 μs
▶ P	Sky	661.39 μs
▶ P	Shadow Gen	409.31 μs
▶ P	Deferred Directional Lighting	362.65 μs
▶ P	Light	293.16 μs
▶ P	Point Light Mask	276.72 μs
▶ P	Fairy Drawing	13.37 μs

Shader Profiler

Provides per-pipeline timing information

Per-line execution cost (iOS and tvOS)

<pre>float3 colorSample = colorMap.sample(colorSampler, in.texCoord.xy*0.15 - uniforms.time[0] * 0.0075).xyz;</pre>	45.71%
<pre>float3 colorSample2 = colorMap.sample(colorSampler, in.texCoord.xy*0.05).xyz;</pre>	1.53%
<pre>float relativeHeight = clamp(in.positionModelSpace.y + 0.5, 0.0, 1.0);</pre>	0.43%
<pre>float2 relativeXZ = clamp((in.positionModelSpace.xz + 1.5) / 3.0, 0.0, 1.0); // Thi should be 1.0 / 2.0</pre>	14.65%
<pre>float spatialVariation = sin(3 * (in.positionModelSpace.x + in.positionModelSpace.z));</pre>	4.27%

Shader Profiler

Provides per-pipeline timing information

Per-line execution cost (iOS and tvOS)

Shader edit and reload

<pre>float3 colorSample = colorMap.sample(colorSampler, in.texCoord.xy*0.15 - uniforms.time[0] * 0.0075).xyz;</pre>	45.71%
<pre>float3 colorSample2 = colorMap.sample(colorSampler, in.texCoord.xy*0.05).xyz;</pre>	1.53%
<pre>float relativeHeight = clamp(in.positionModelSpace.y + 0.5, 0.0, 1.0);</pre>	0.43%
<pre>float2 relativeXZ = clamp((in.positionModelSpace.xz + 1.5) / 3.0, 0.0, 1.0); // Thi should be 1.0 / 2.0</pre>	14.65%
<pre>float spatialVariation = sin(3 * (in.positionModelSpace.x + in.positionModelSpace.z));</pre>	4.27%

Shader Profiler

Provides per-pipeline timing information

Per-line execution cost (iOS and tvOS)

Shader edit and reload

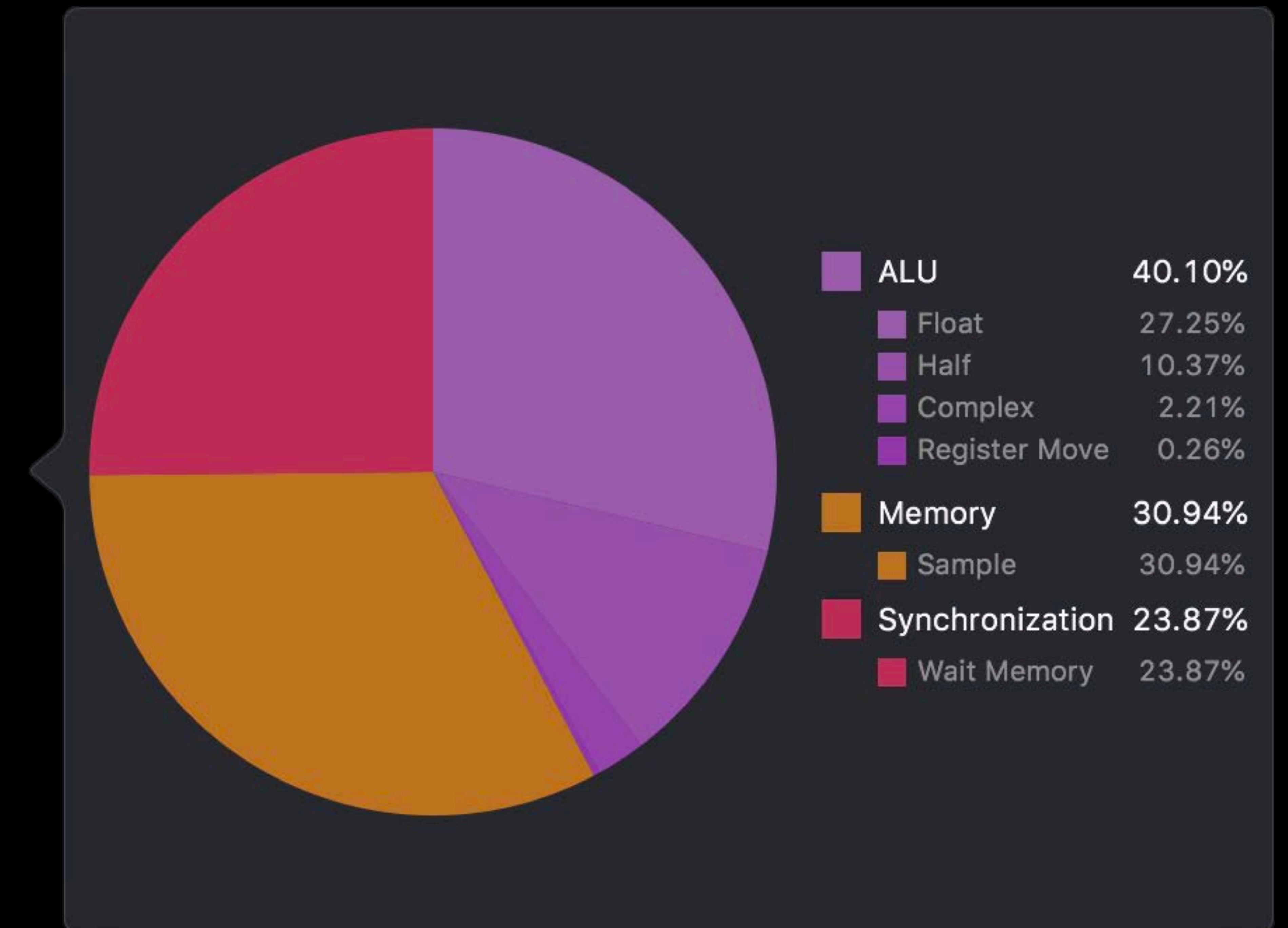
Get into shader debugger

<pre>float3 colorSample = colorMap.sample(colorSampler, in.texCoord.xy*0.15 - uniforms.time[0] * 0.0075).xyz;</pre>	45.71%
<pre>float3 colorSample2 = colorMap.sample(colorSampler, in.texCoord.xy*0.05).xyz;</pre>	1.53%
<pre>float relativeHeight = clamp(in.positionModelSpace.y + 0.5, 0.0, 1.0);</pre>	0.43%
<pre>float2 relativeXZ = clamp((in.positionModelSpace.xz + 1.5) / 3.0, 0.0, 1.0); // Thi should be 1.0 / 2.0</pre>	14.65%
<pre>float spatialVariation = sin(3 * (in.positionModelSpace.x + in.positionModelSpace.z));</pre>	4.27%

Enhanced Shader Profiler for A11

NEW

Instruction category cost breakdown per line

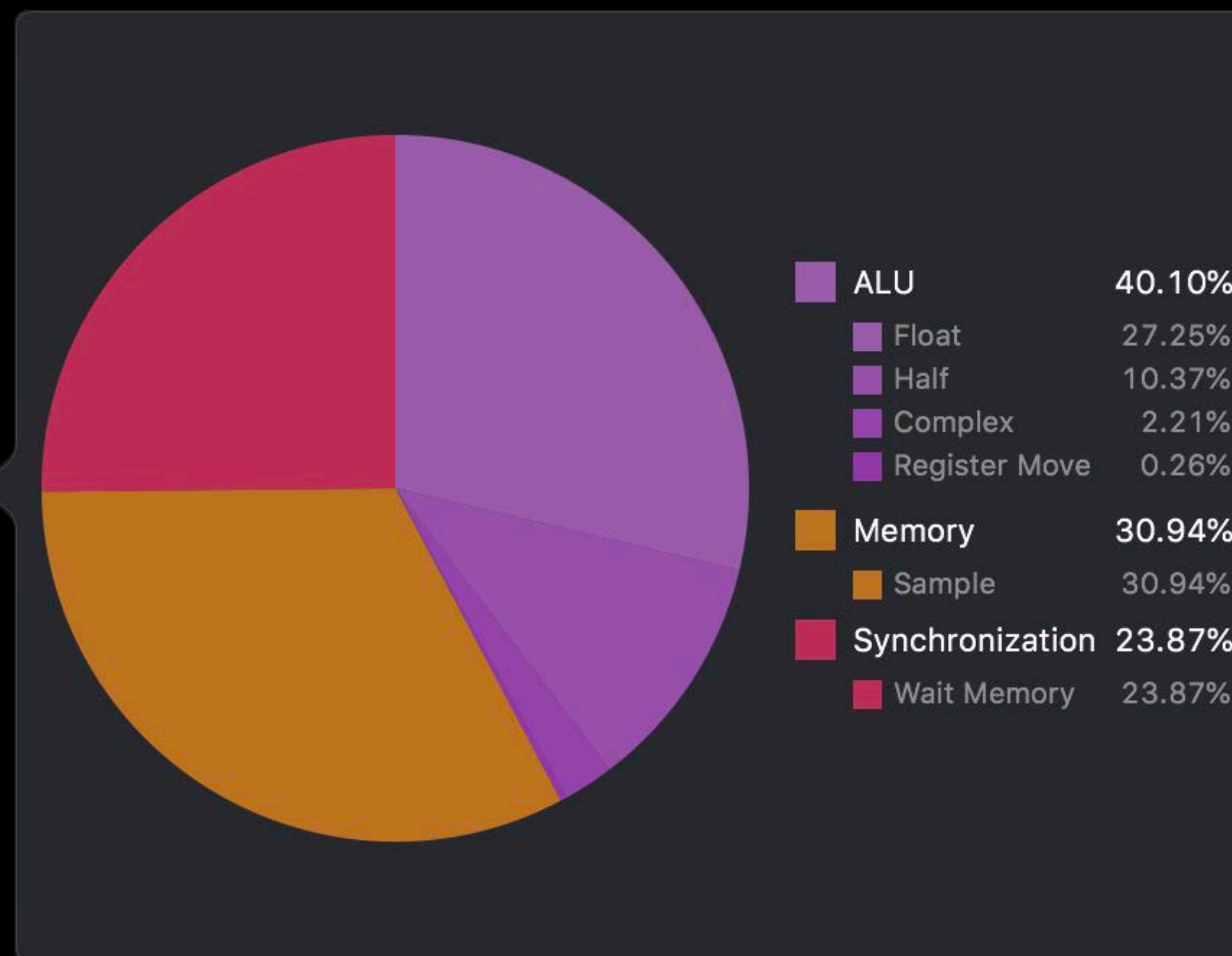


Enhanced Shader Profiler for A11

NEW

Instruction category cost breakdown per line

- ALU — Float, half, and complex
- Memory — Sample, load, and store operations

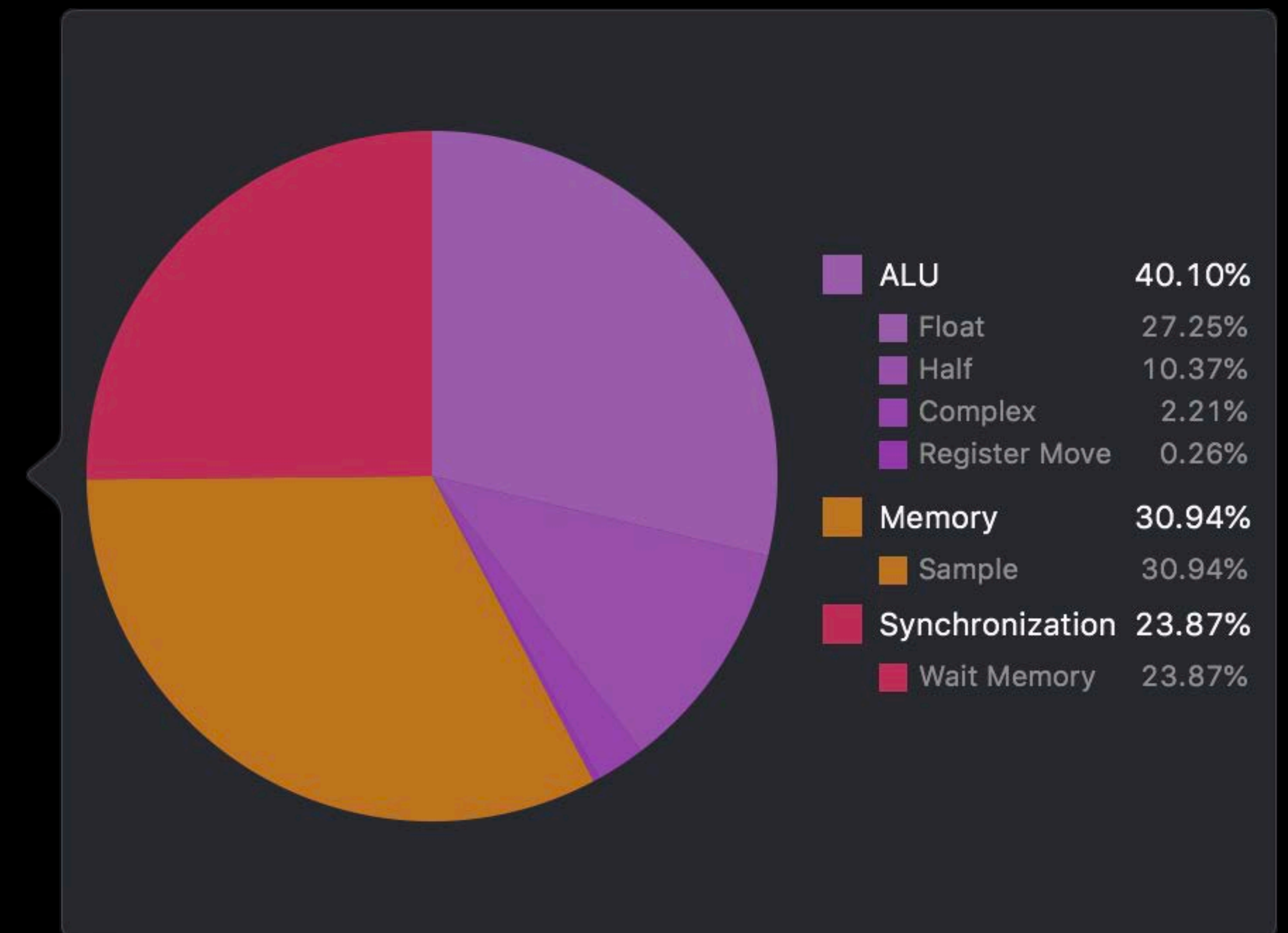


Enhanced Shader Profiler for A11

NEW

Instruction category cost breakdown per line

- ALU — Float, half, and complex
- Memory — Sample, load, and store operations
- Synchronization — Wait memory, barriers, or atomics



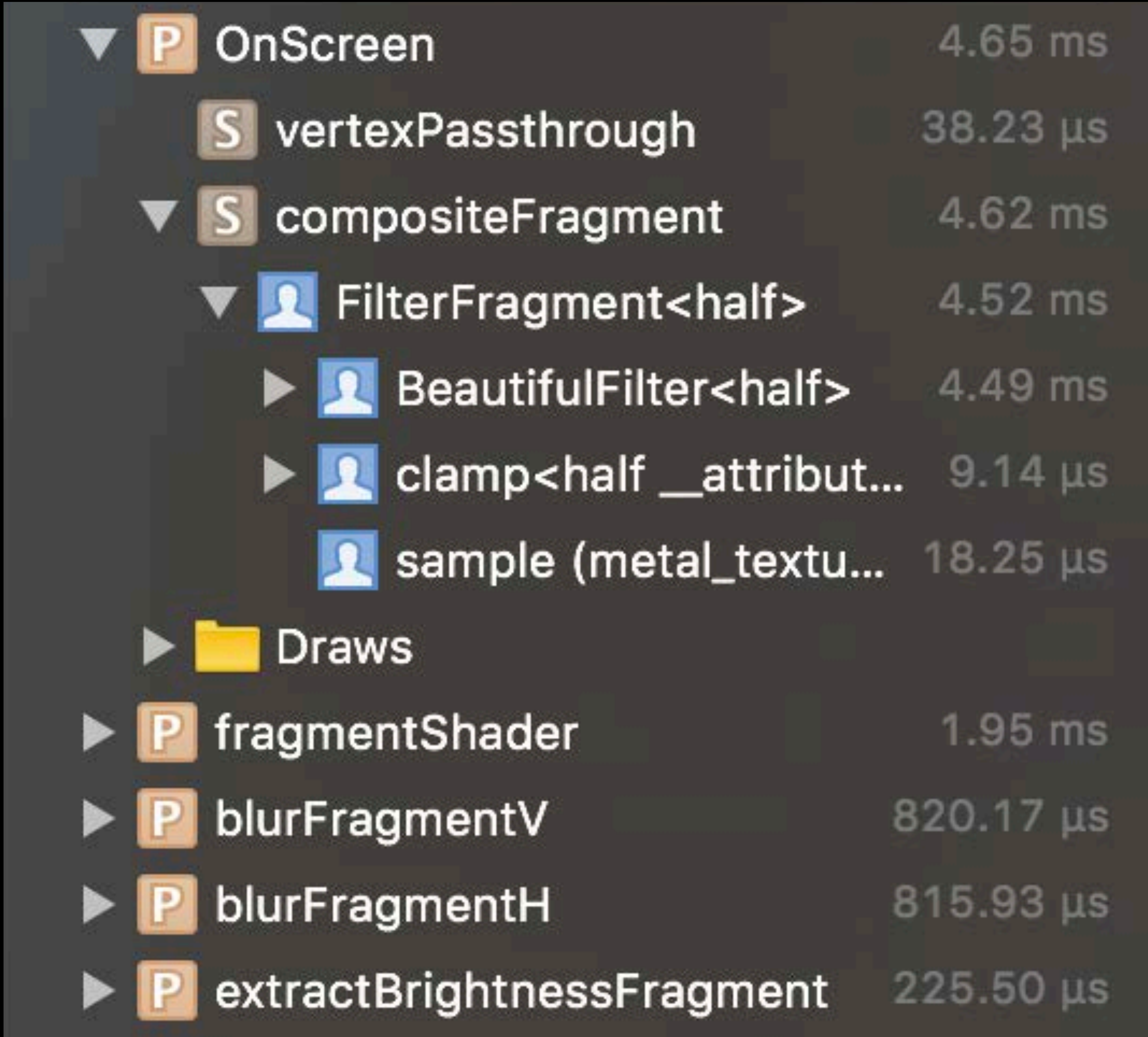
Enhanced Shader Profiler for A11

NEW

Instruction category cost breakdown per line

- ALU — Float, half, and complex
- Memory — Sample, load, and store operations
- Synchronization — Wait memory, barriers, or atomics

Visibility into inline function cost



The screenshot displays a hierarchical view of shader costs. The root node is 'OnScreen' (4.65 ms), which is expanded to show 'vertexPassthrough' (38.23 μs) and 'compositeFragment' (4.62 ms). 'compositeFragment' is further expanded to show 'FilterFragment<half>' (4.52 ms), which is expanded to show 'BeautifulFilter<half>' (4.49 ms), 'clamp<half __attribut...' (9.14 μs), and 'sample (metal_textu...' (18.25 μs). Below these are 'Draws' (yellow folder icon), 'fragmentShader' (1.95 ms), 'blurFragmentV' (820.17 μs), 'blurFragmentH' (815.93 μs), and 'extractBrightnessFragment' (225.50 μs). Each node has a small icon (P for Program, S for Shader, or a person icon for Filter) and a right-aligned time value.

▼ P	OnScreen	4.65 ms
	S	vertexPassthrough 38.23 μs
▼ S	compositeFragment	4.62 ms
▼	FilterFragment<half>	4.52 ms
▶	BeautifulFilter<half>	4.49 ms
▶	clamp<half __attribut...	9.14 μs
▶	sample (metal_textu...	18.25 μs
▶	Draws	
▶ P	fragmentShader	1.95 ms
▶ P	blurFragmentV	820.17 μs
▶ P	blurFragmentH	815.93 μs
▶ P	extractBrightnessFragment	225.50 μs

Demo

Max Christ

Access to Shader Sources

NEW

New Metal Compiler option

Xcode project

- "Yes, include source code" from build settings

Command line

- "-MO" compiler option

▼ Metal Compiler - Build Options	
Setting	
Enable fast math	No ◊
Header Search Paths	
Ignore Warnings	No ◊
Metal language revision	◊
▶ Optimization Level	◊
Other Metal Compiler Flags	
Preprocessor Definitions	
▼ Produce debugging information	
	<Multiple values> ◊
Debug	Yes, include source code ◊
Release	No ◊
Treat Warnings as Errors	No ◊

Recap

Geometry viewer

Recap

Geometry viewer

Shader debugger

Recap

Geometry viewer

Shader debugger

Enhanced shader profiler

More Information

<https://developer.apple.com/wwdc18/608>

Metal Game Performance Optimization

Hall 1

Fri 10:00 AM

Metal Debugging and Profiling Lab

Technology Lab 5

Fri 12:00 PM

 **WWDC18**