#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

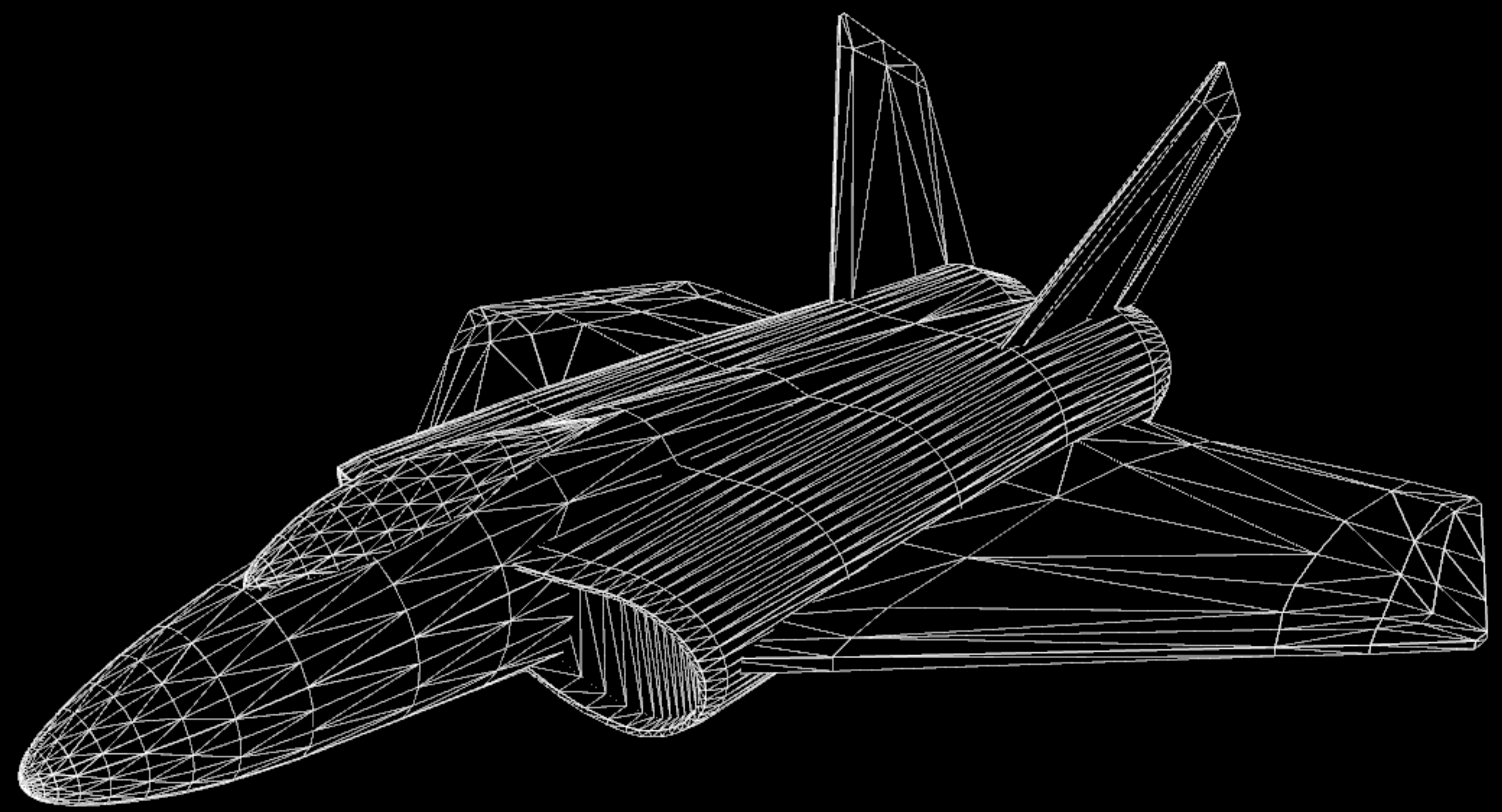
Investigating Vertex Issues with Geometry Viewer

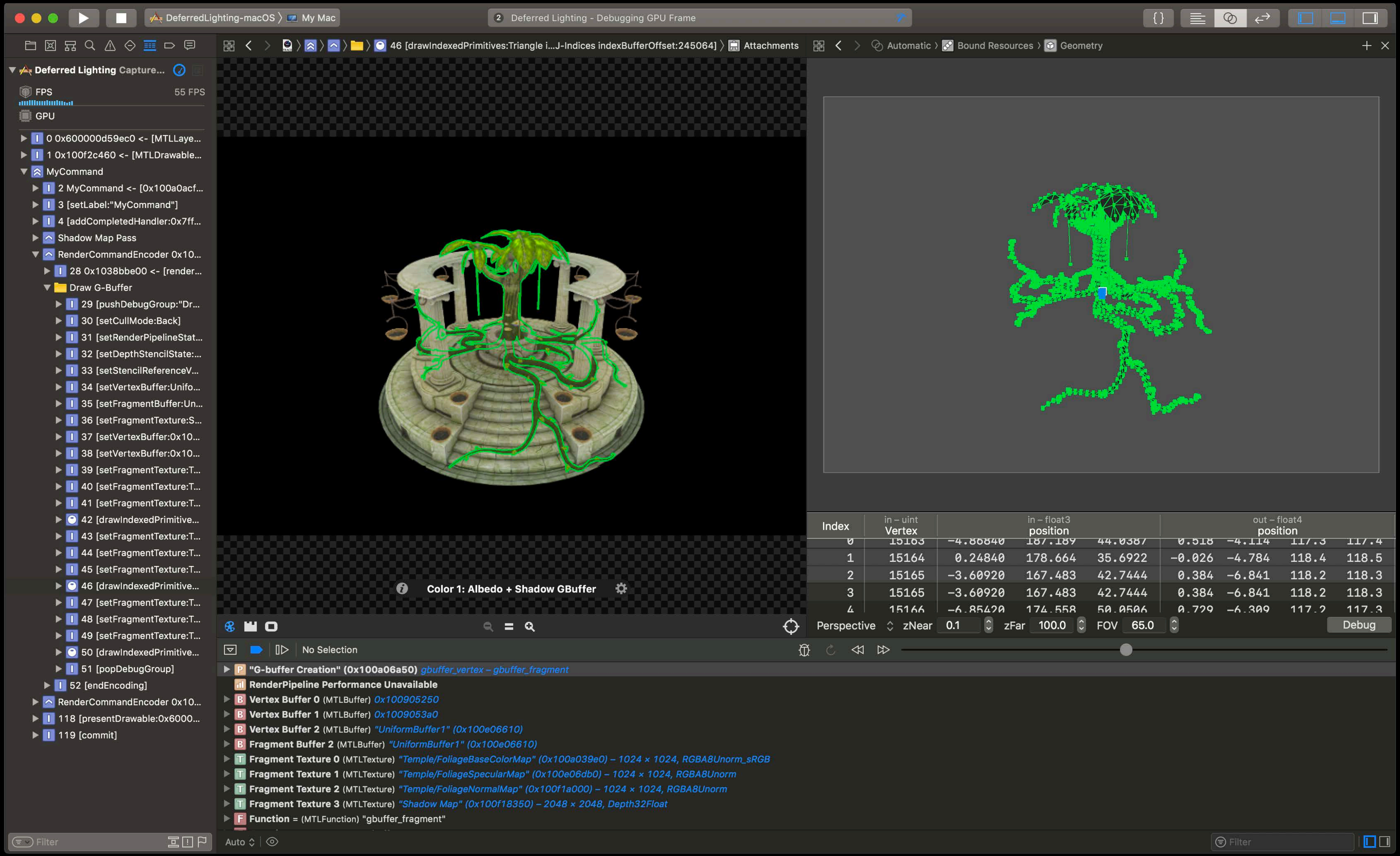
Things to Watch in Vertex Stage

Vertex inputs

Indices

Output





DeferredLighting-macOS 〉 My Mac
Deferred Lighting - Debugging GPU Frame
46 [drawIndexedPrimitives:Triangle i...J-Indices indexBufferOffset:245064] 〉 Attachments
Automatic 〉 Bound Resources 〉 Geometry
Deferred Lighting Capture...
FPS 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]
Filter
Color 1: Albedo + Shadow GBuffer
Index
in – uint Vertex
in – float3 position
out – float4 position
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
Perspective
zNear 0.1
zFar 100.0
FOV 65.0
Debug
No Selection
"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 × 1024, RGBA8Unorm_sRGB
Fragment Texture 1 (MTLTexture) "Temple/FoliageSpecularMap" (0x100e06db0) – 1024 × 1024, RGBA8Unorm
Fragment Texture 2 (MTLTexture) "Temple/FoliageNormalMap" (0x100f1a000) – 1024 × 1024, RGBA8Unorm
Fragment Texture 3 (MTLTexture) "Shadow Map" (0x100f18350) – 2048 × 2048, Depth32Float
Function = (MTLFunction) "gbuffer_fragment"
Auto
Filter

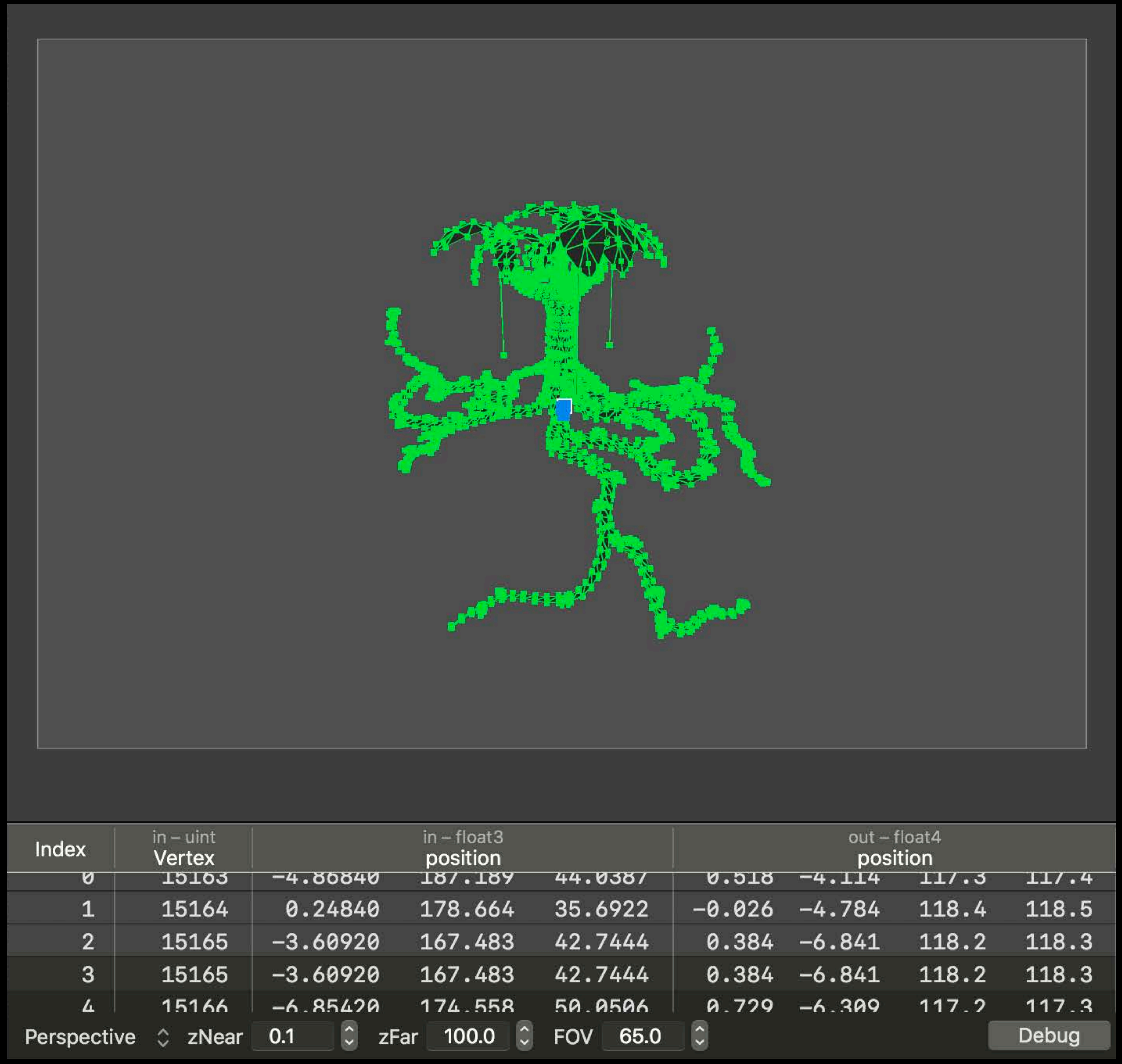
Geometry Viewer

NEW

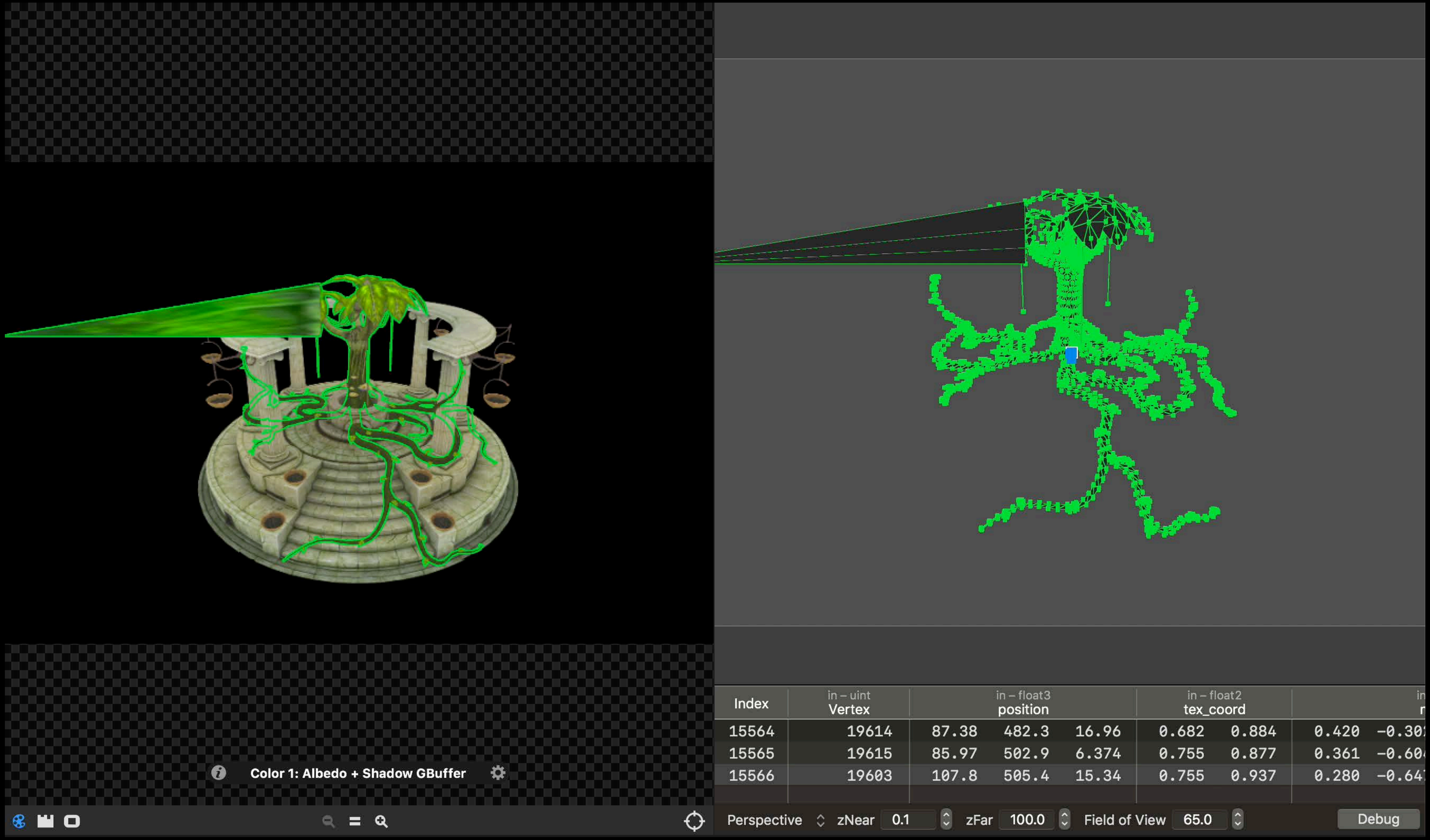
Visualize the post-vertex transform data in 3D

Access to all vertex data

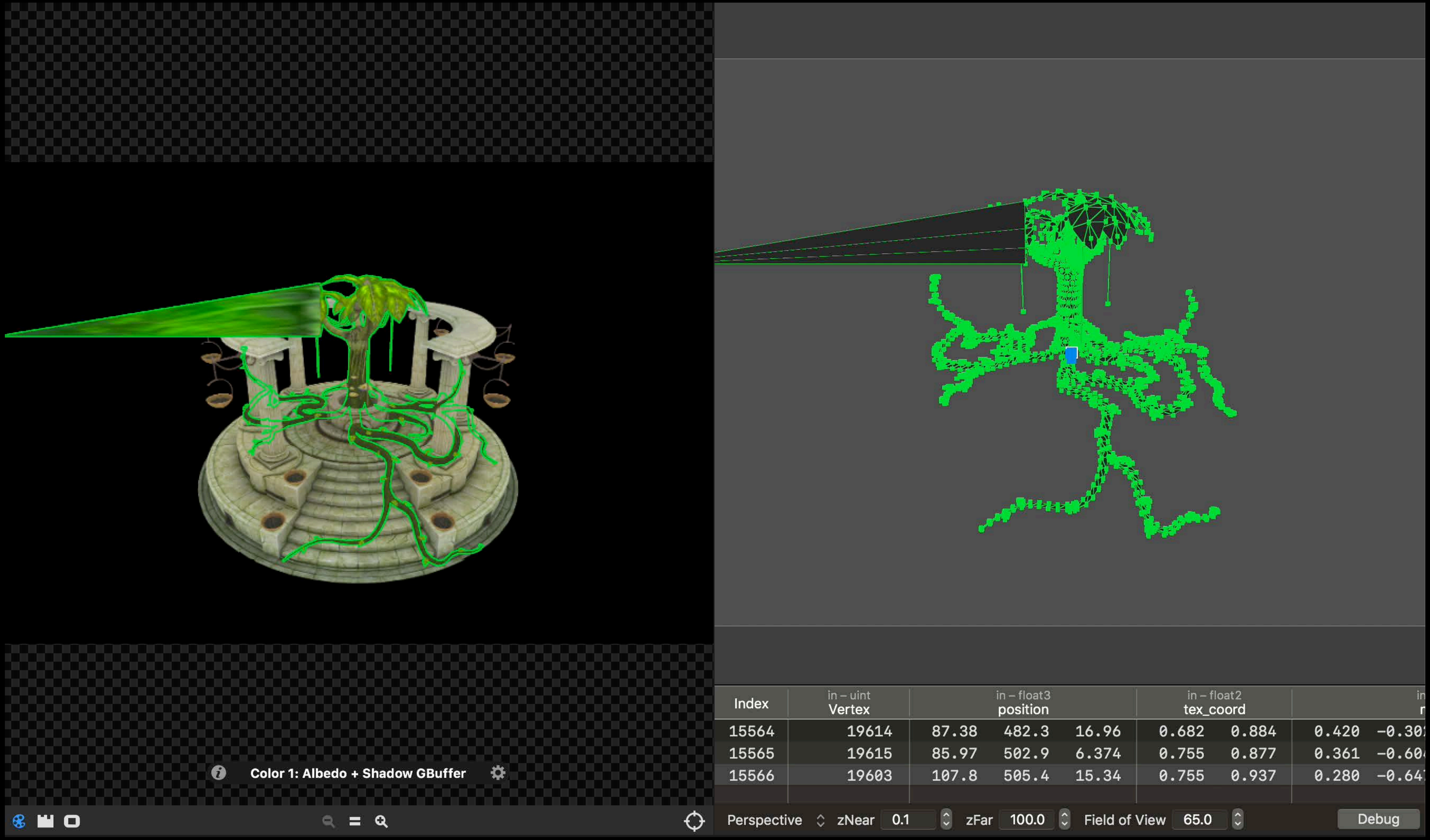
Per-draw call view



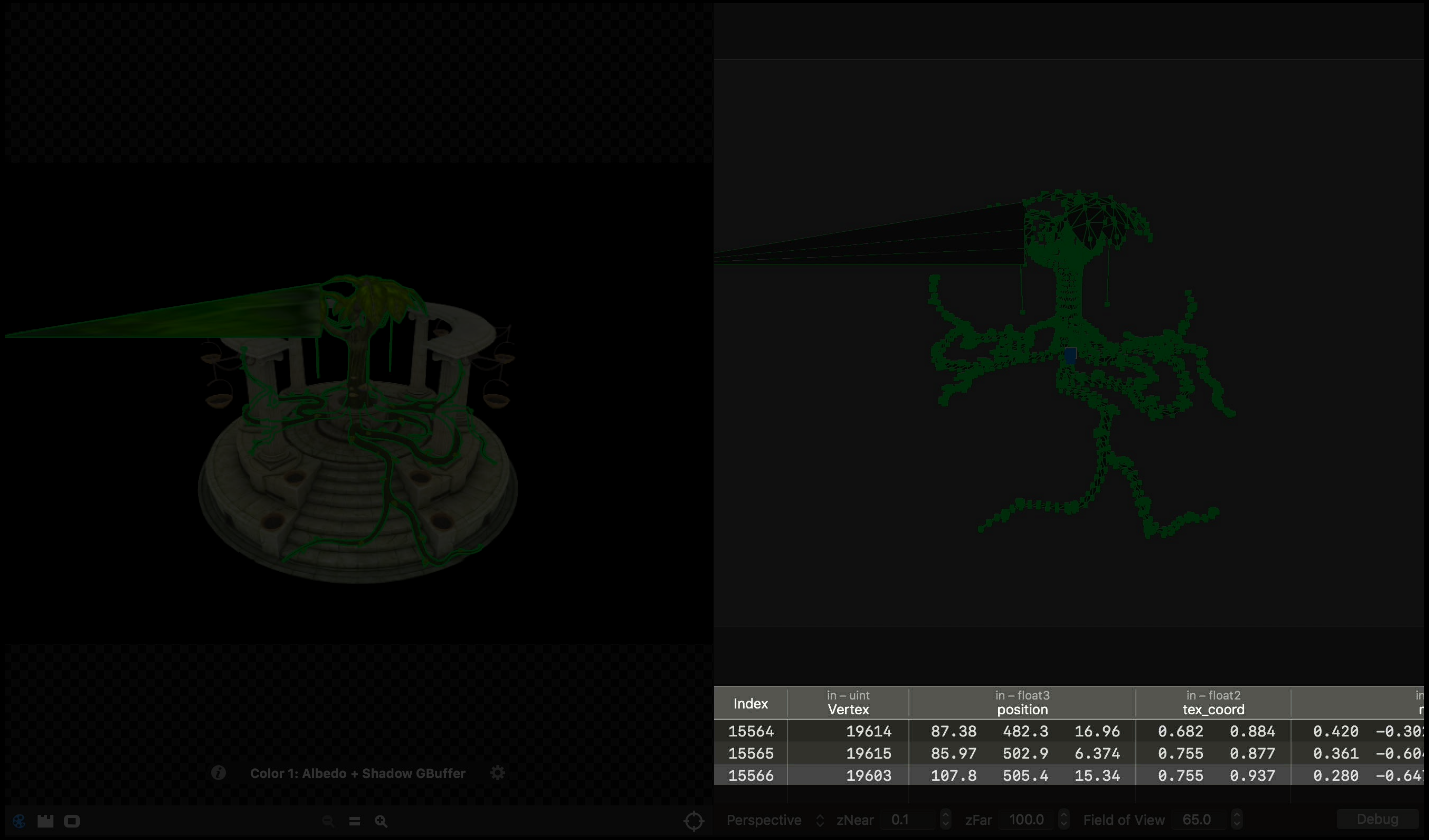
Visibly Wrong Triangles



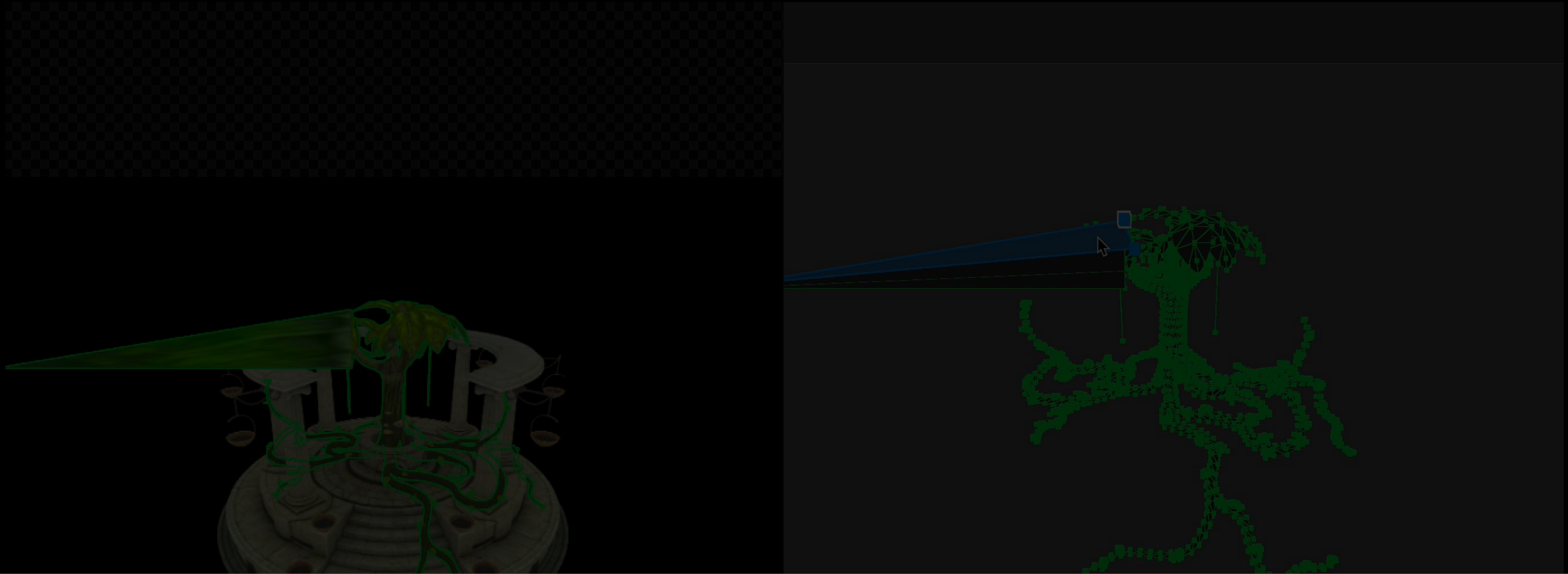
Visibly Wrong Triangles



Visibly Wrong Triangles

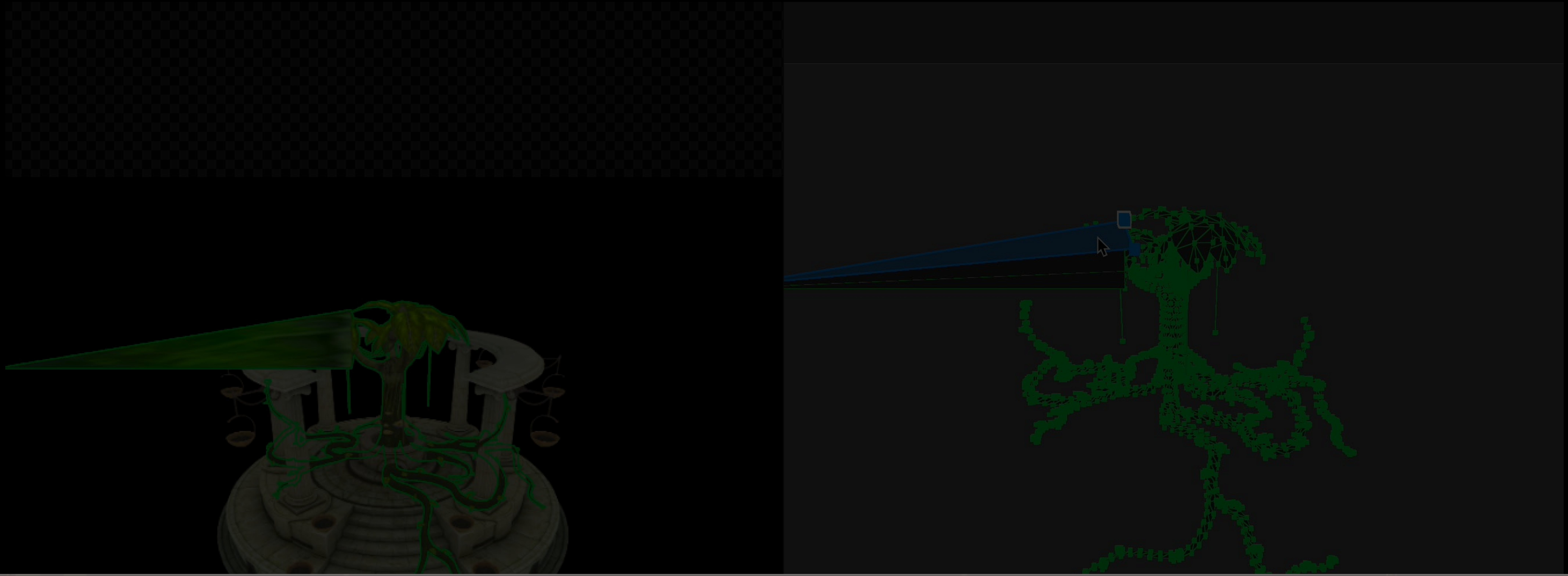


Visibly Wrong Triangles



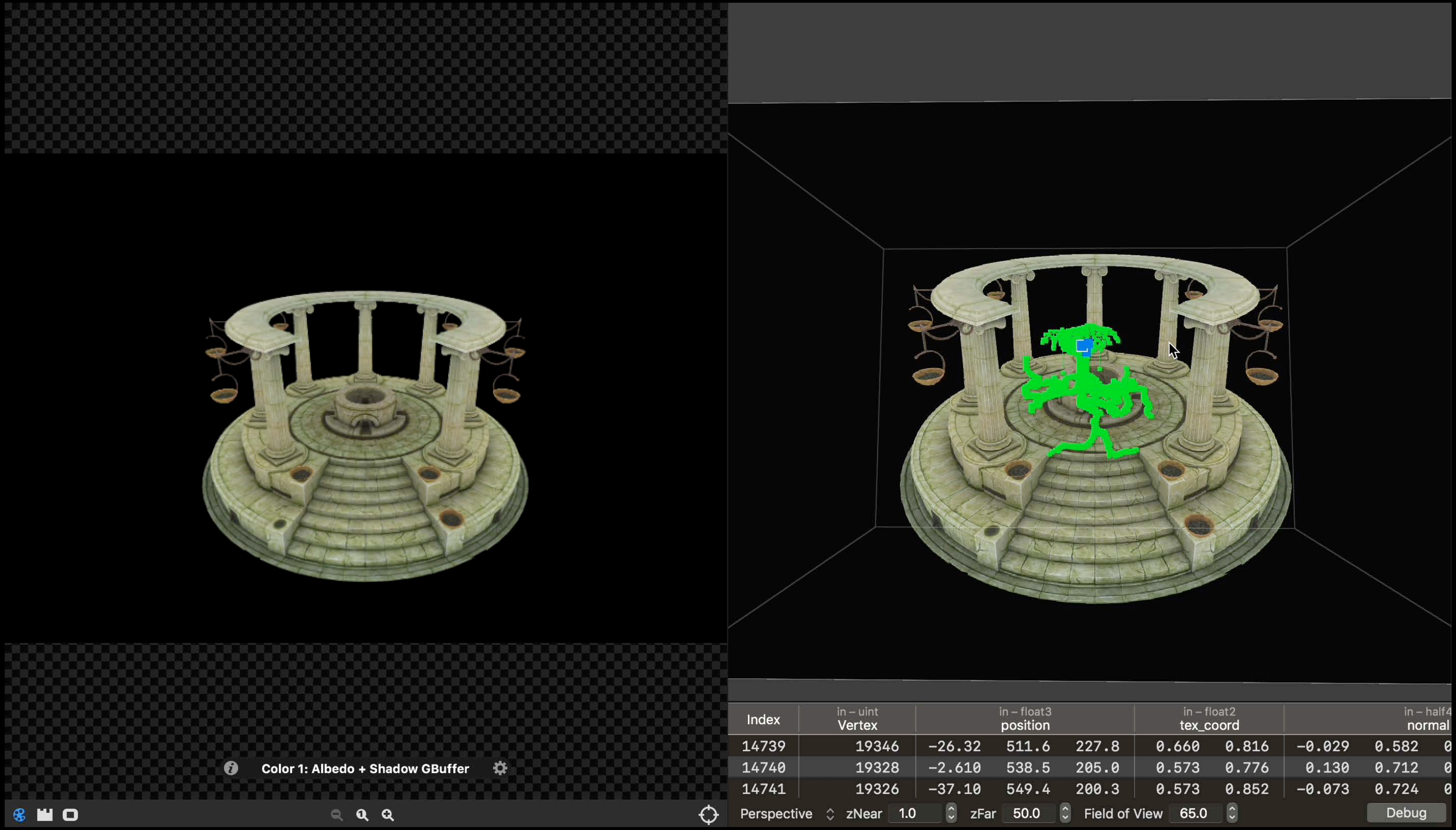
Index	in – uint Vertex	in – float3 position			in – float2 tex_coord		in r	
15564	19614	87.38	482.3	16.96	0.682	0.884	0.420	−0.30
15565	19615	85.97	502.9	6.374	0.755	0.877	0.361	−0.60
15566	19603	107.8	505.4	15.34	0.755	0.937	0.280	−0.64

Visibly Wrong Triangles

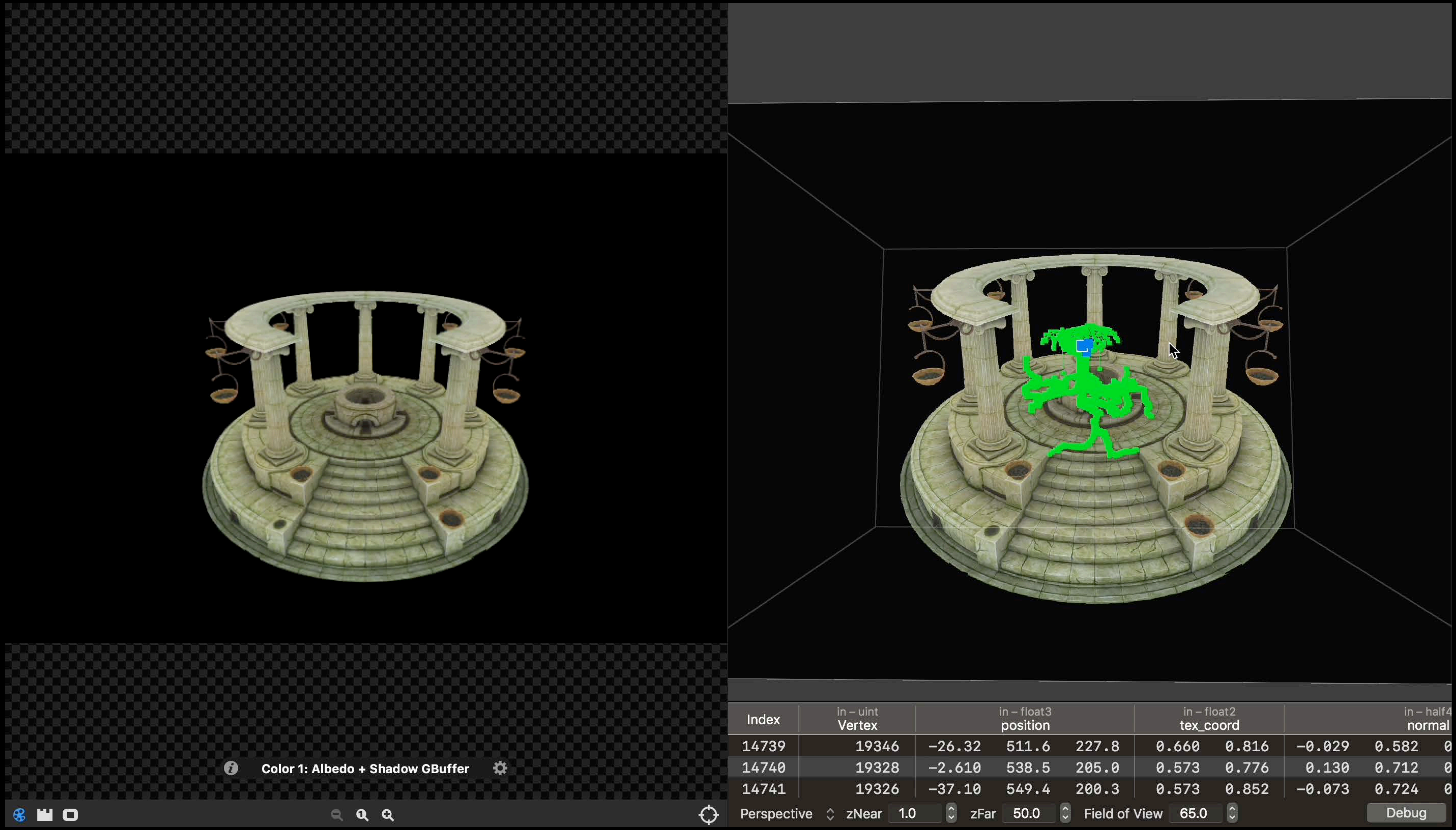


Index	in – uint Vertex	in – float3 position			in – float2 tex_coord		in n	
15564	19614	87.38	482.3	16.96	0.682	0.884	0.420	−0.30
15565	19615	85.97	502.9	6.374	0.755	0.877	0.361	−0.60
15566	19603	107.8	505.4	15.34	0.755	0.937	0.280	−0.64

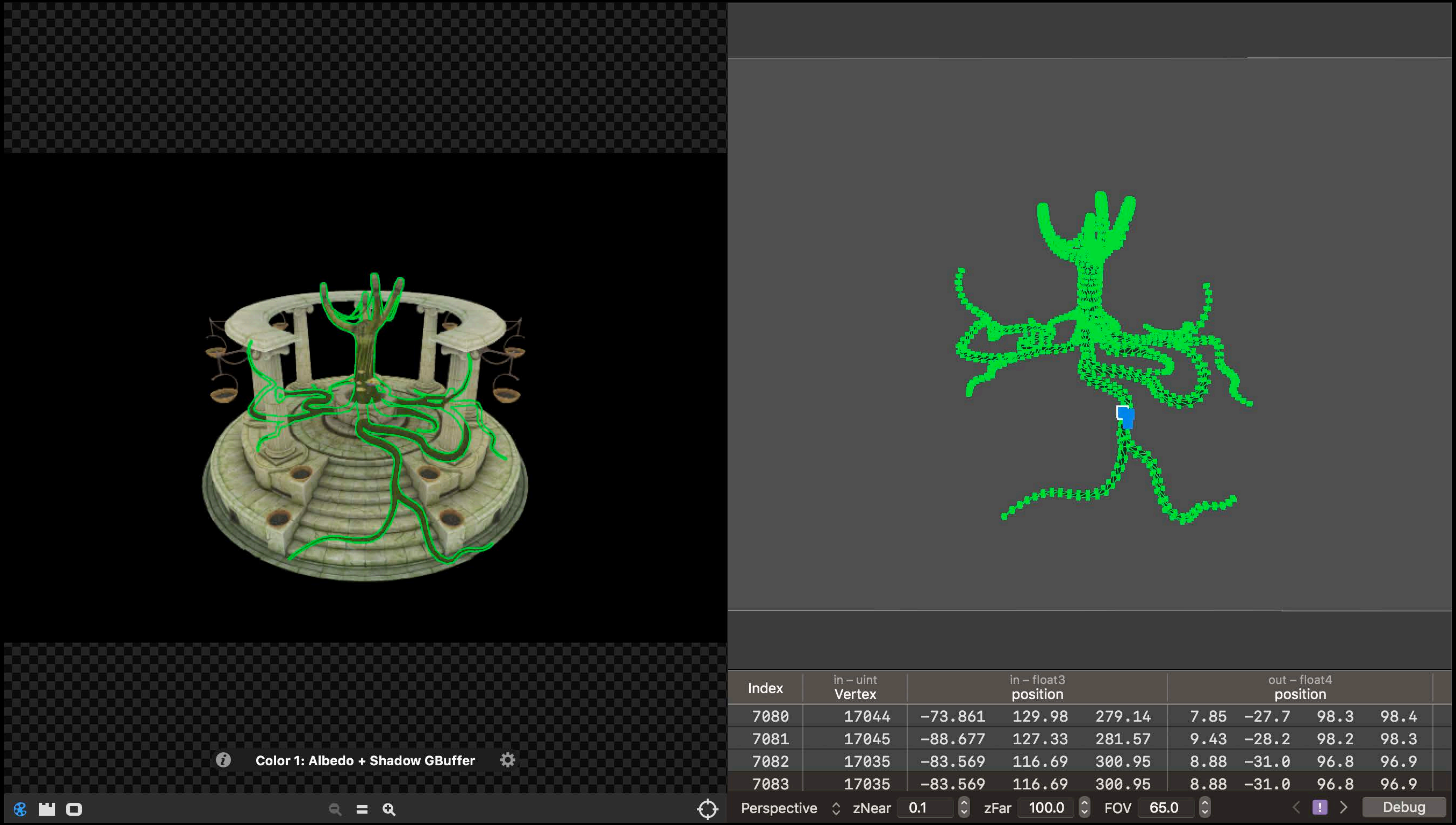
Out of Frustum Objects



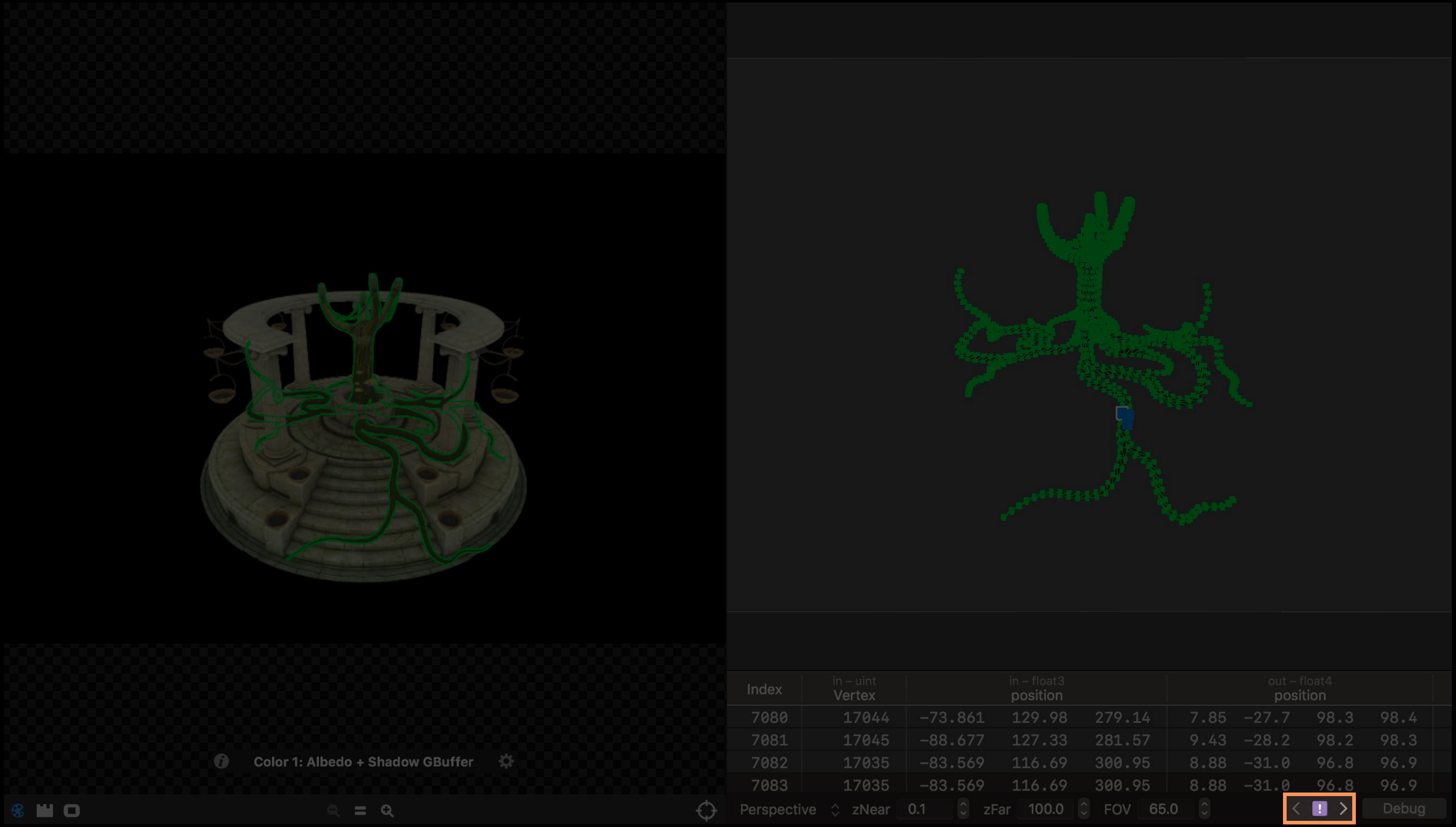
Out of Frustum Objects



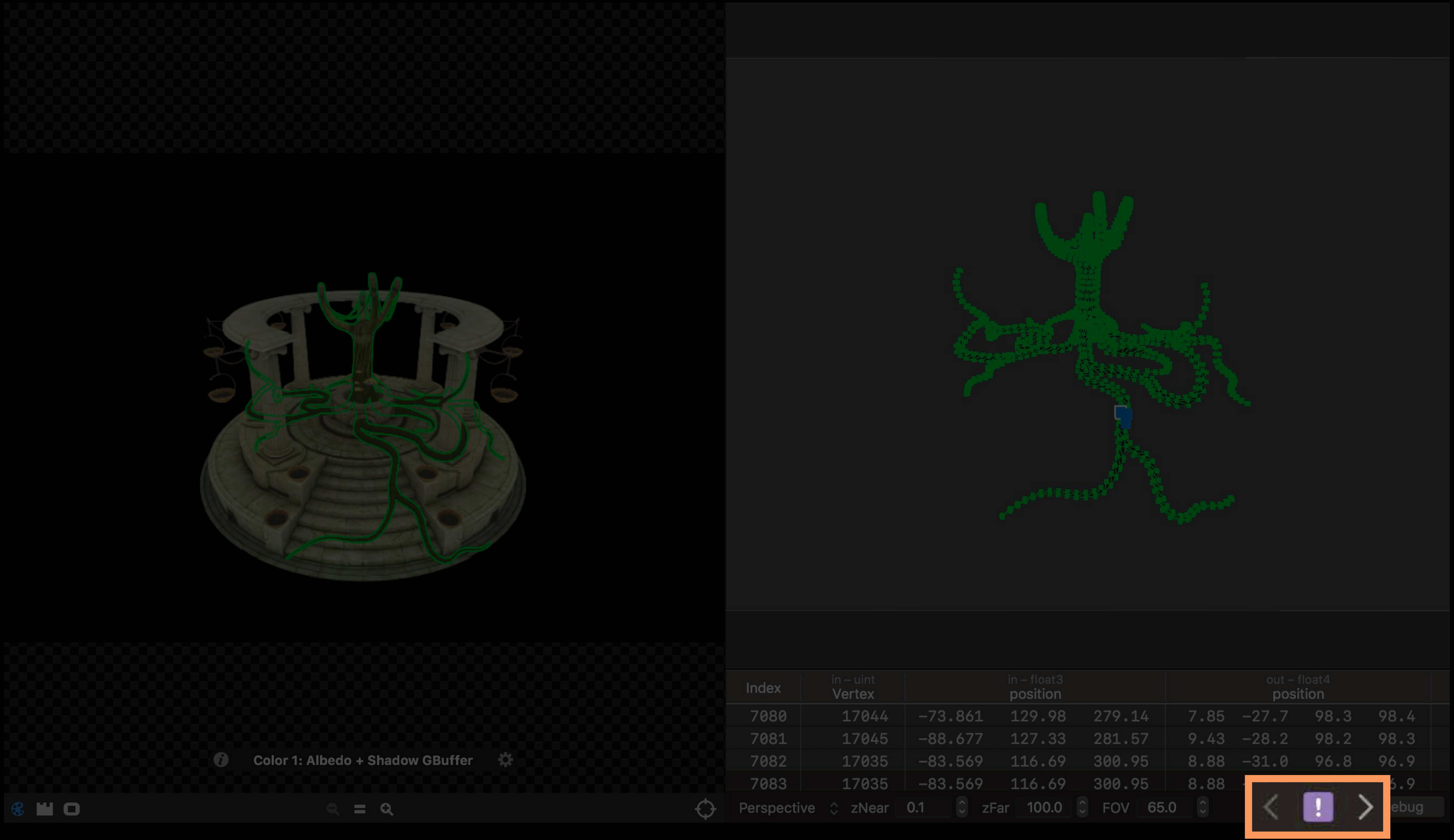
Missing Triangles



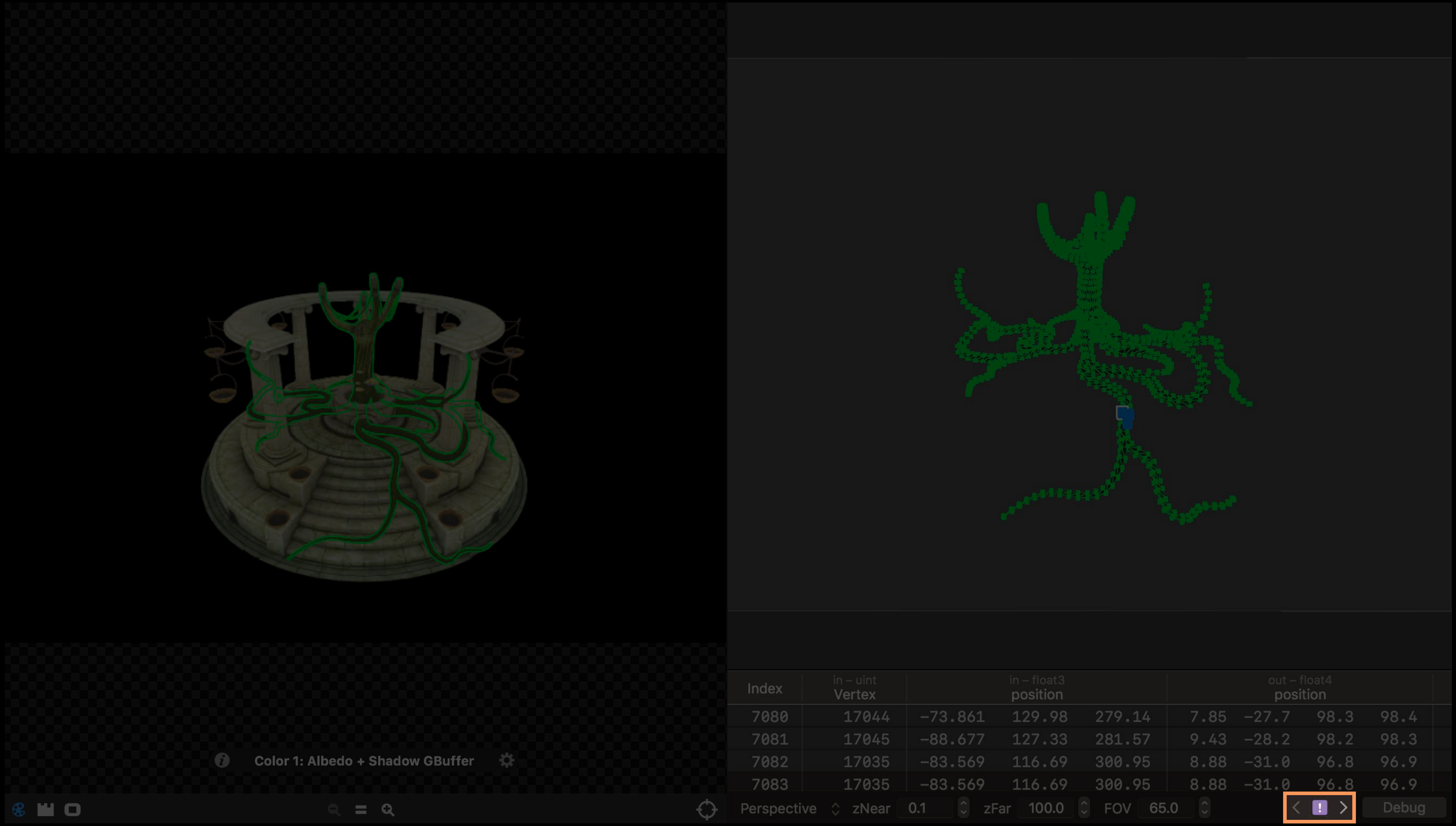
Missing Triangles



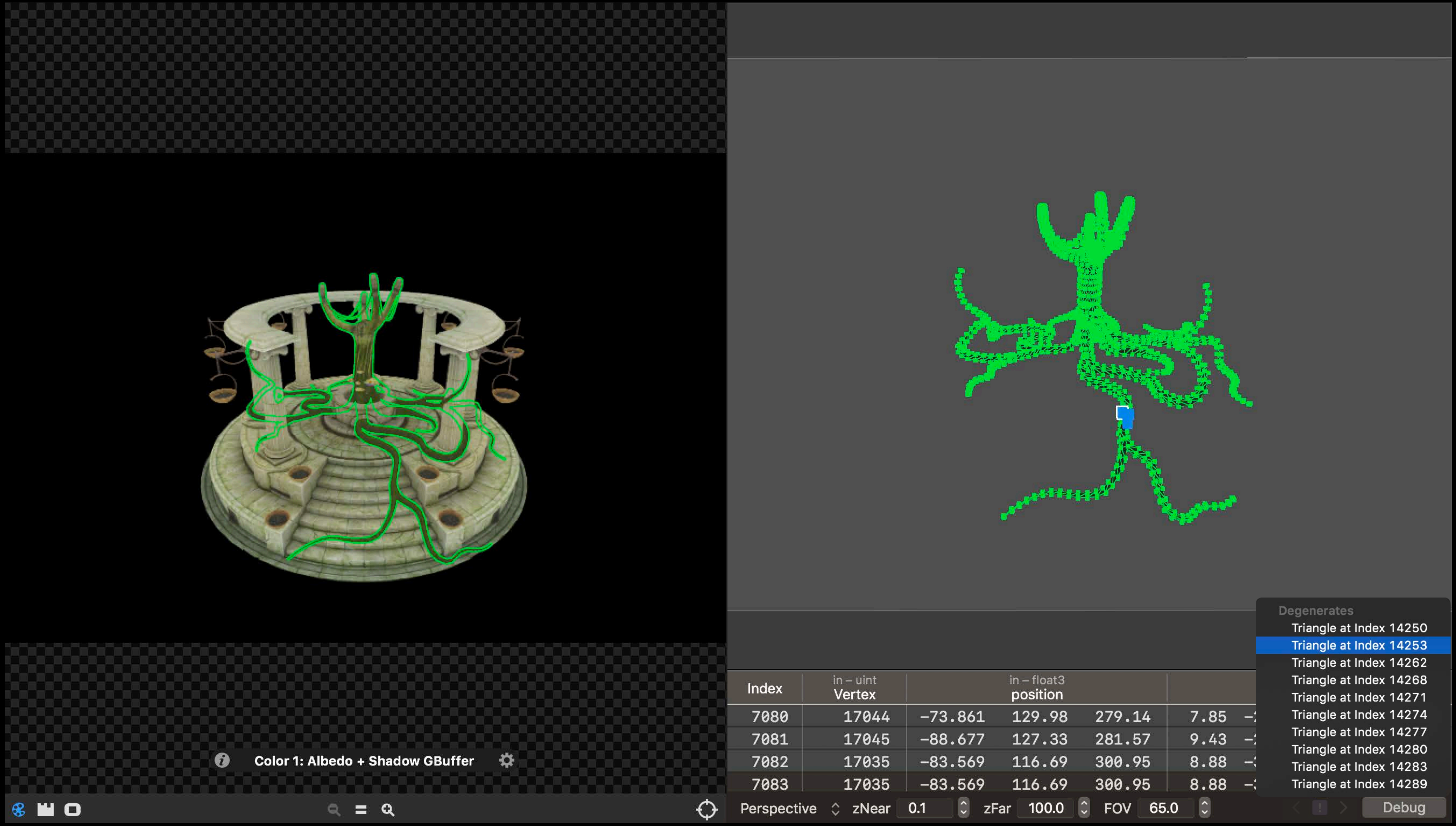
Missing Triangles



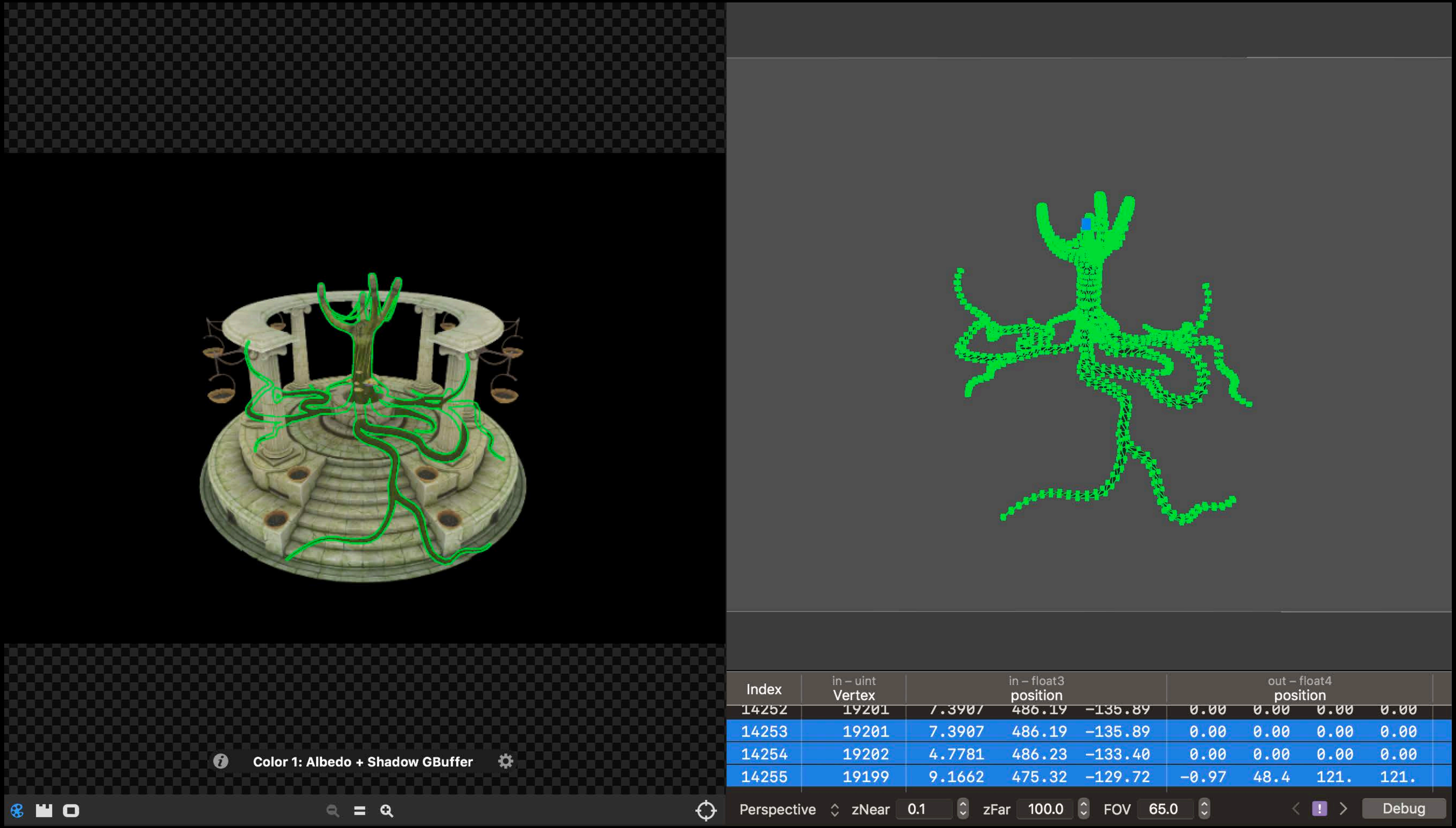
Missing Triangles



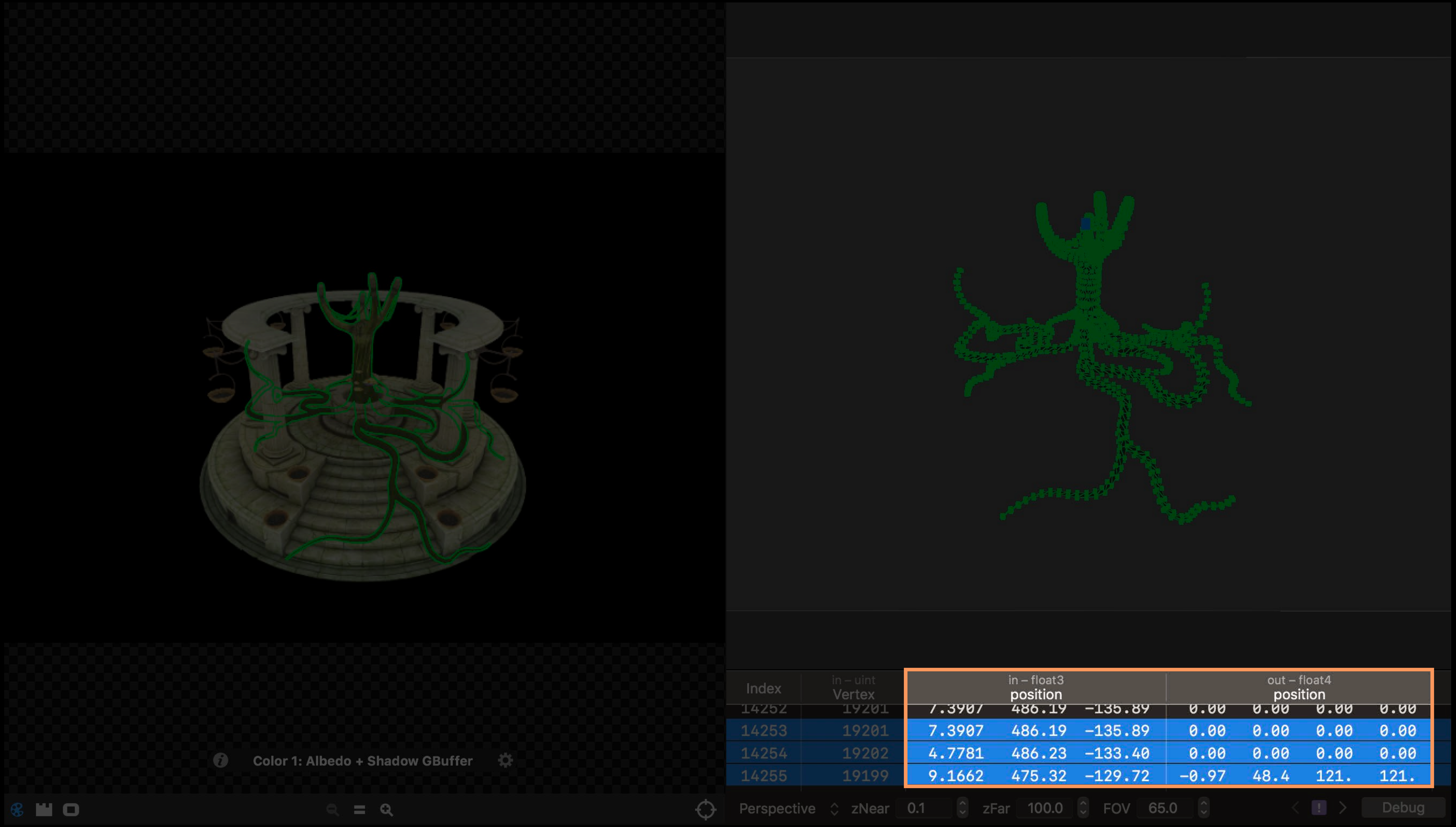
Missing Triangles



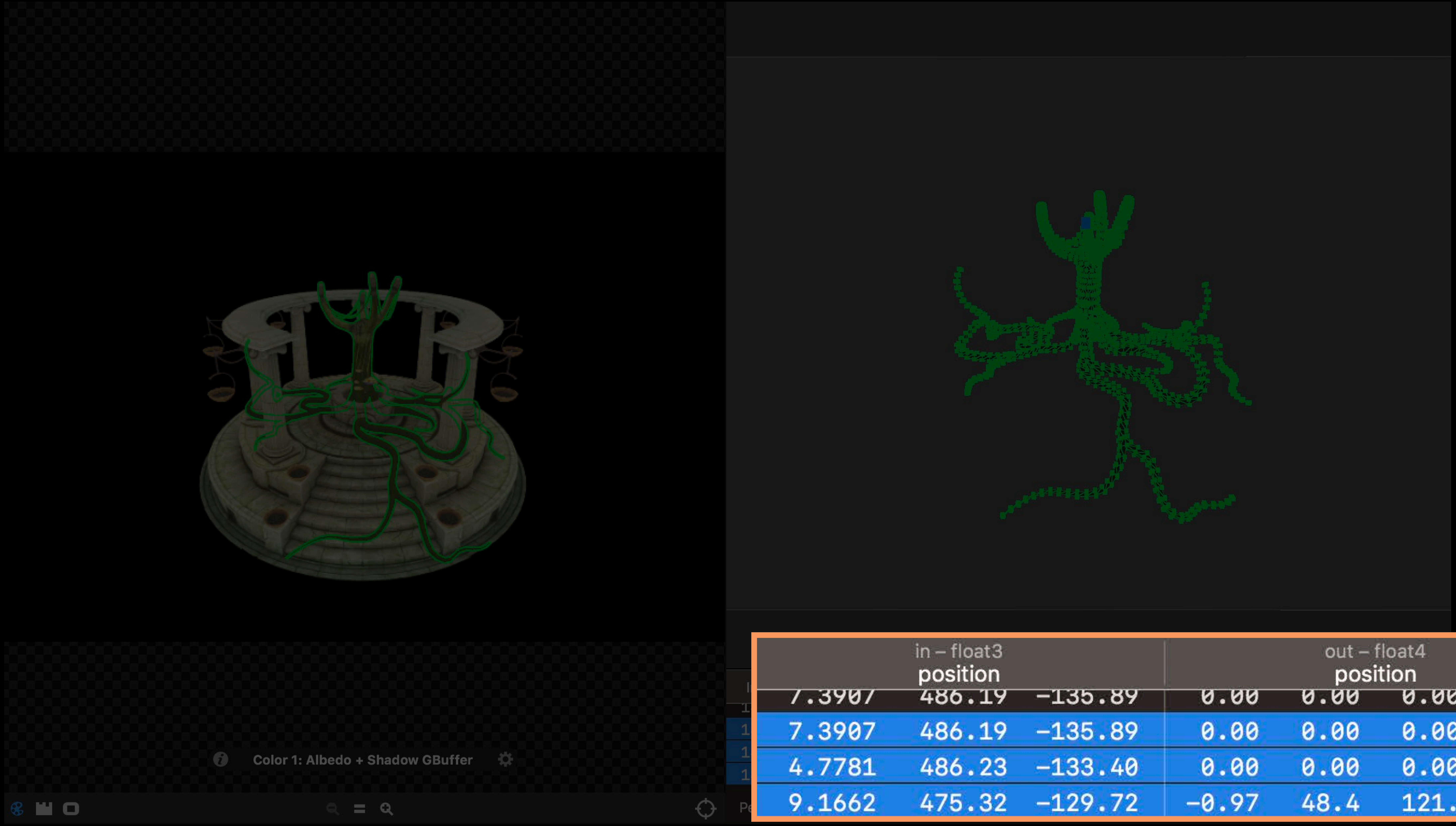
Missing Triangles



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

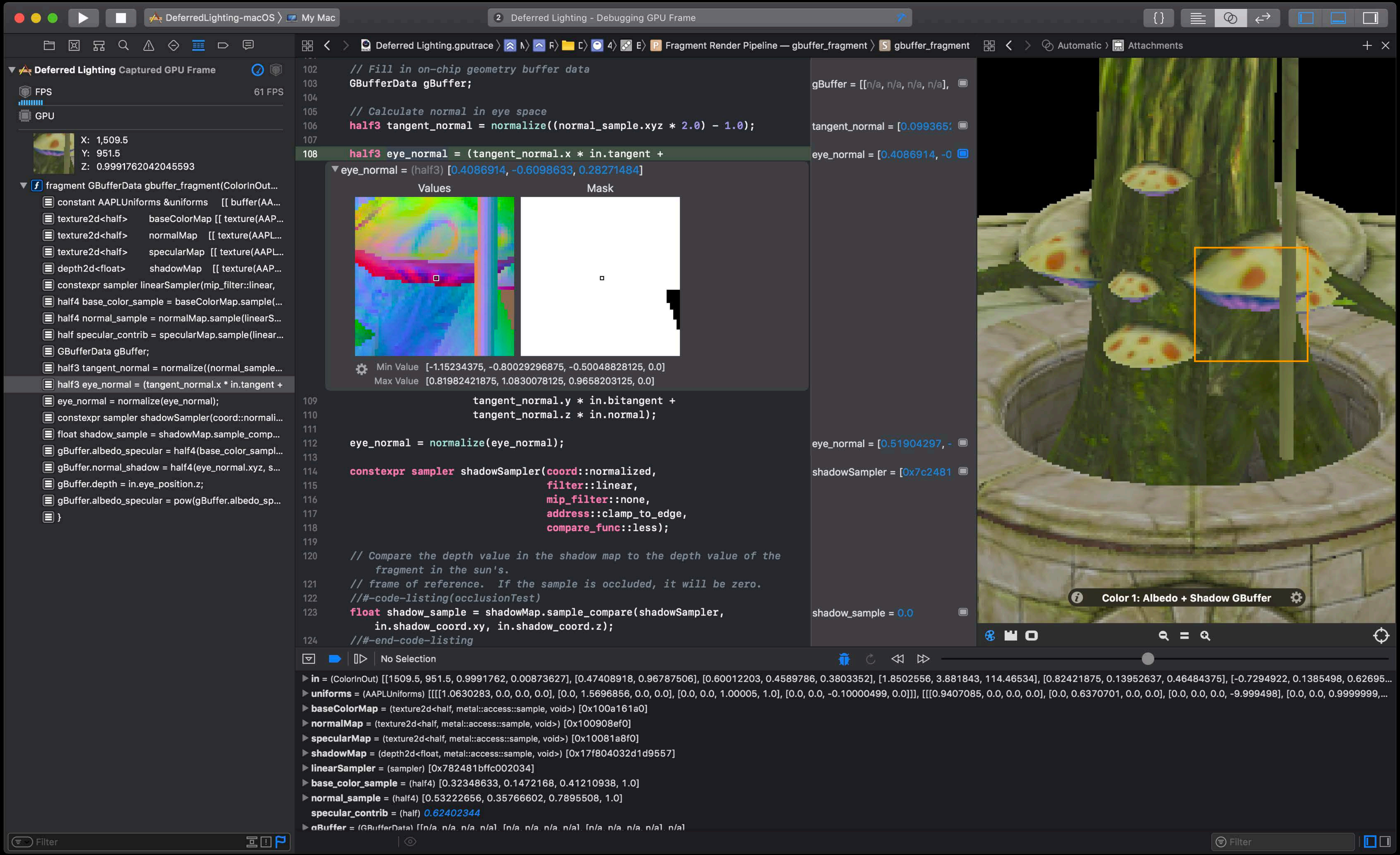
Shader Debugging

Math heavy code

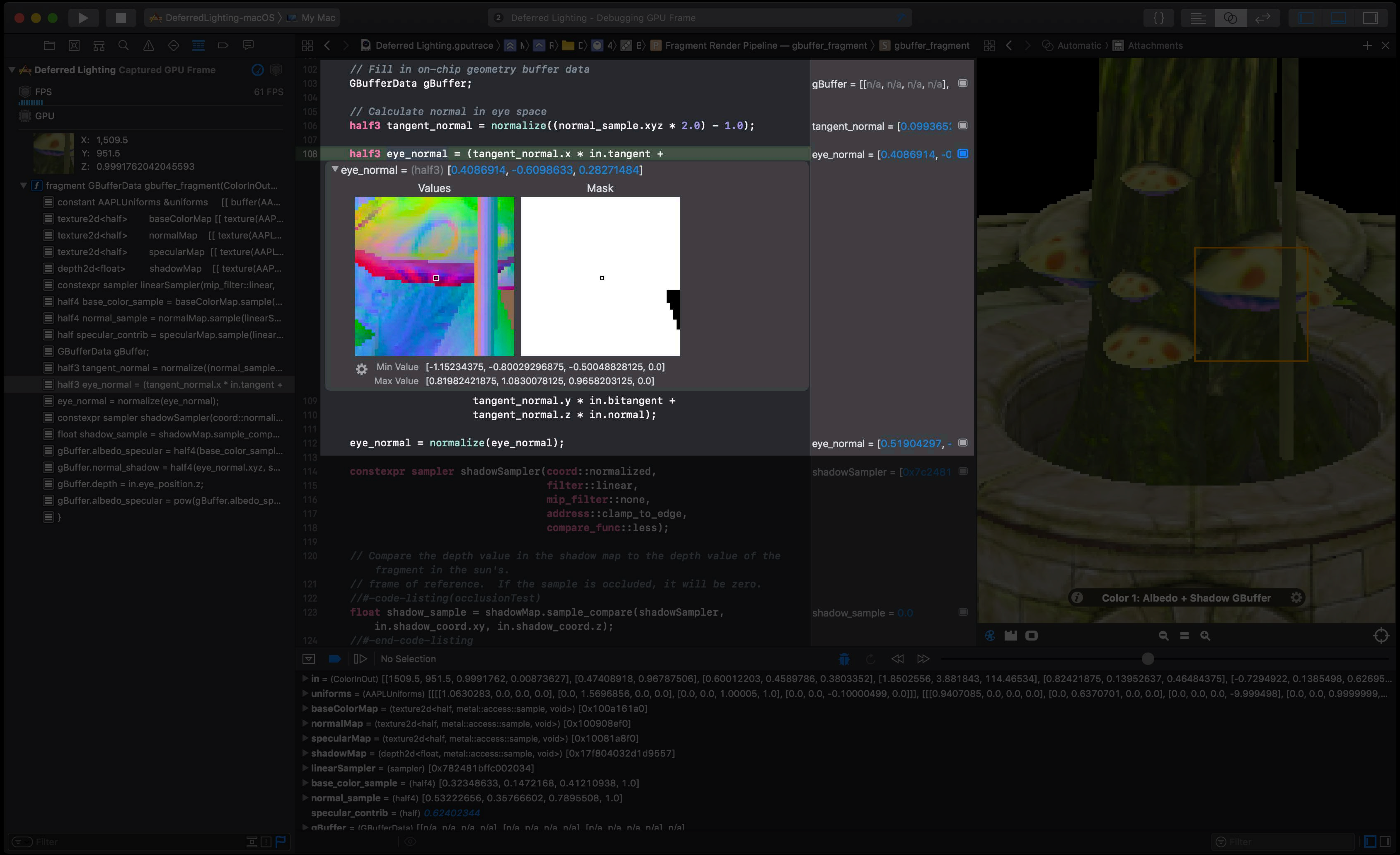
Highly parallel

Unity's "Book of the Dead"

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



DeferredLighting-macOS
My Mac
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...
}
Deferred Lighting.gputrace
Fragment Render Pipeline — gbuffer_fragment
gbuffer_fragment
Automatic
Attachments
// 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]
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 = 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
eye_normal = [0.51904297, -
shadowSampler = [0x7c2481
shadow_sample = 0.0
Color 1: Albedo + Shadow GBuffer
No Selection
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]
Filter
Filter



DeferredLighting-macOS
My Mac
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 +
eye_normal = (half3) [0.4086914, -0.6098633, 0.28271484]
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 = 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
eye_normal = [0.51904297, -
shadowSampler = [0x7c2481
shadow_sample = 0.0
Color 1: Albedo + Shadow GBuffer
No Selection
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
Filter

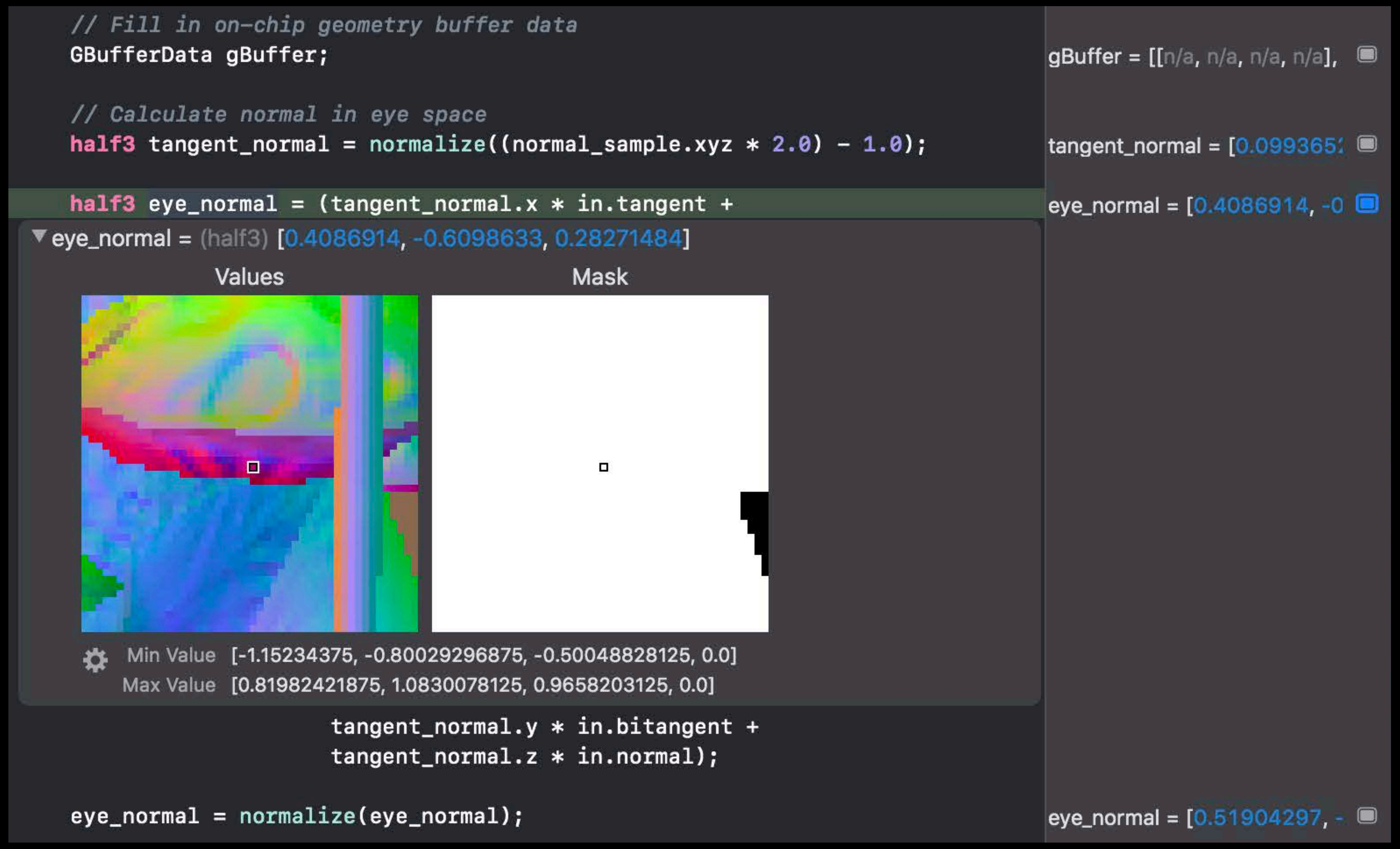
Shader Debugger

NEW

New tool for debugging Metal shaders

Rich variable visualization across thousands of threads

Real data from GPU



Shader Debugger

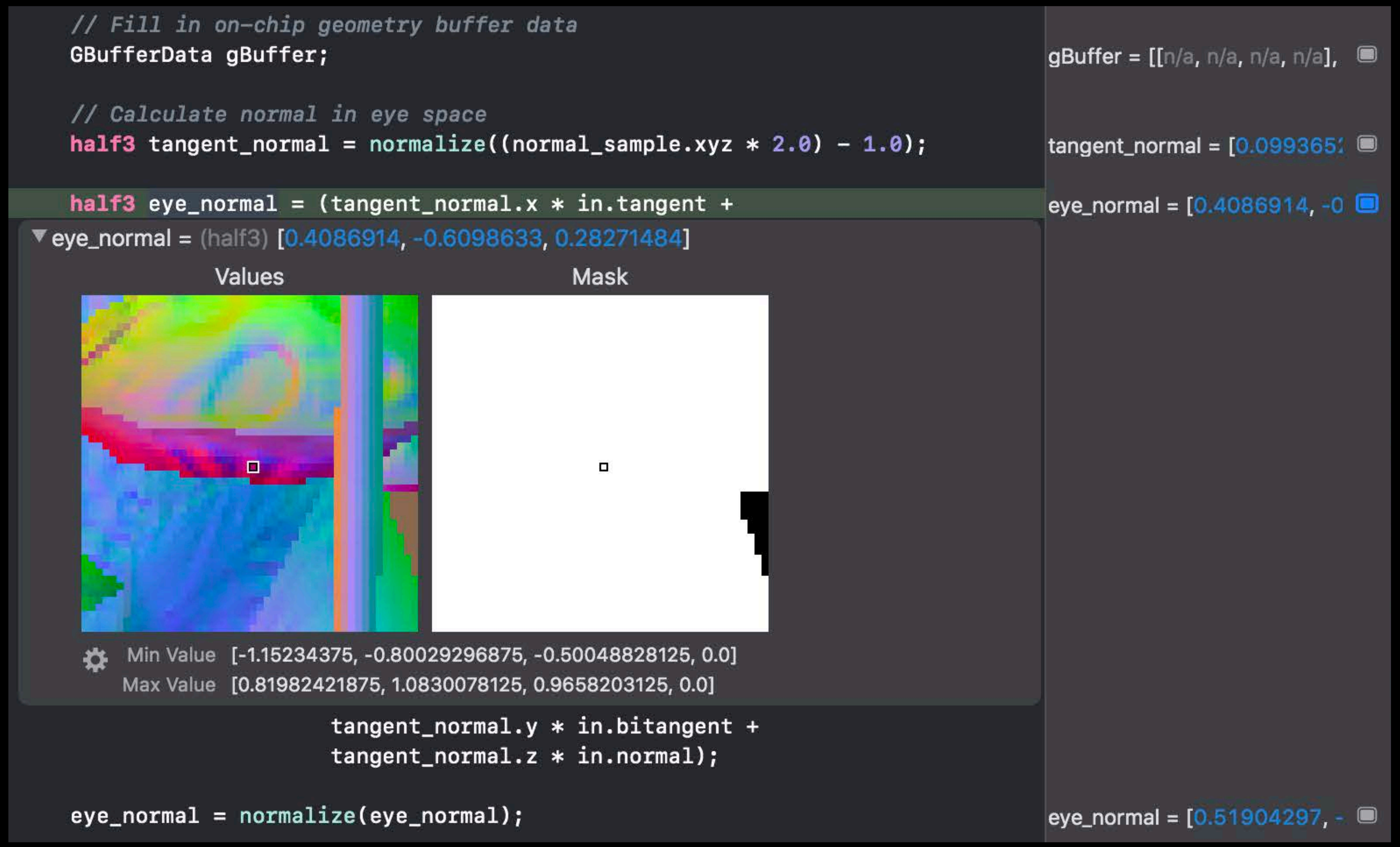
NEW

New tool for debugging Metal shaders

Rich variable visualization across thousands of threads

Real data from GPU

Flexible stepping



Shader Debugger

NEW

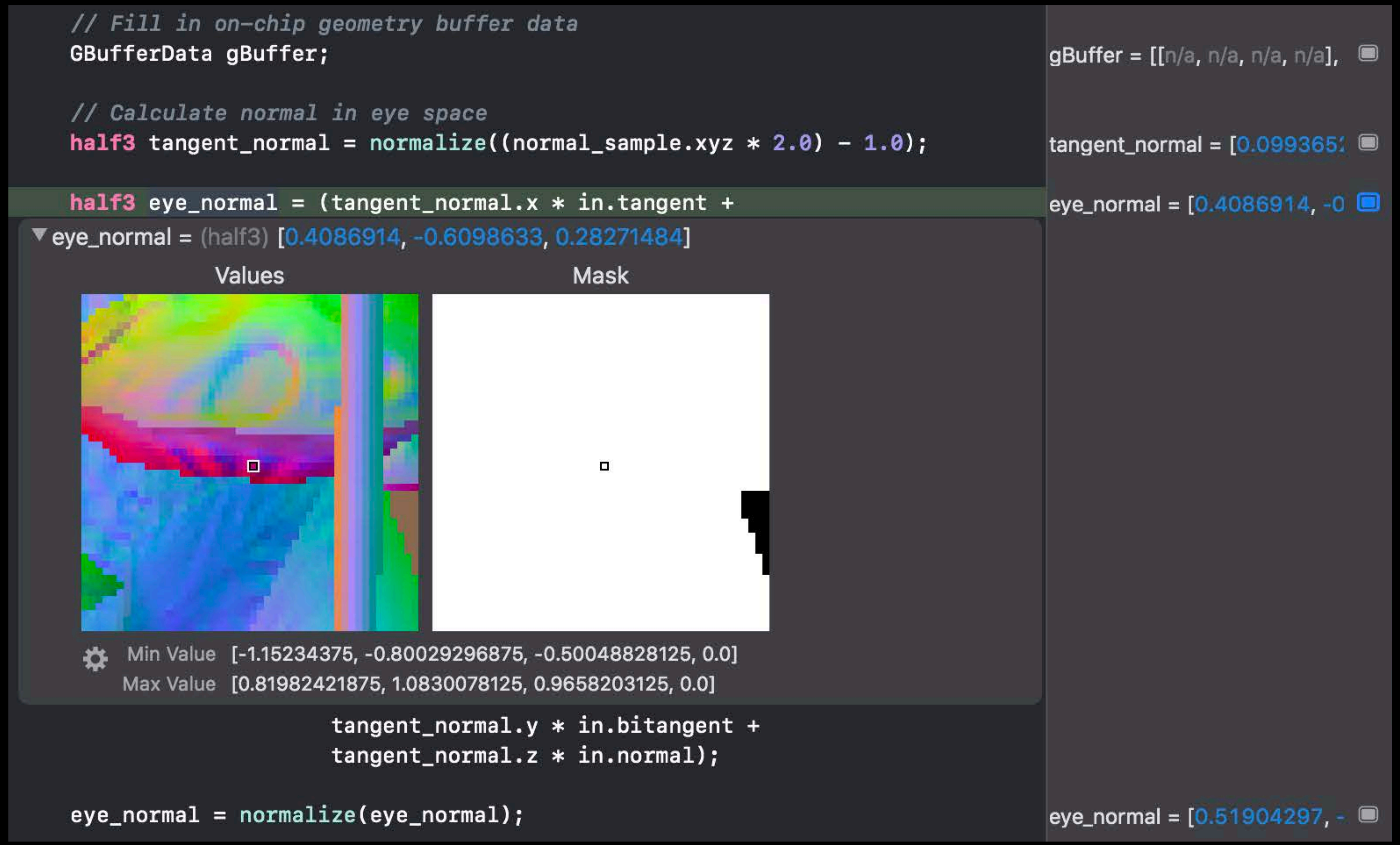
New tool for debugging Metal shaders

Rich variable visualization across thousands of threads

Real data from GPU

Flexible stepping

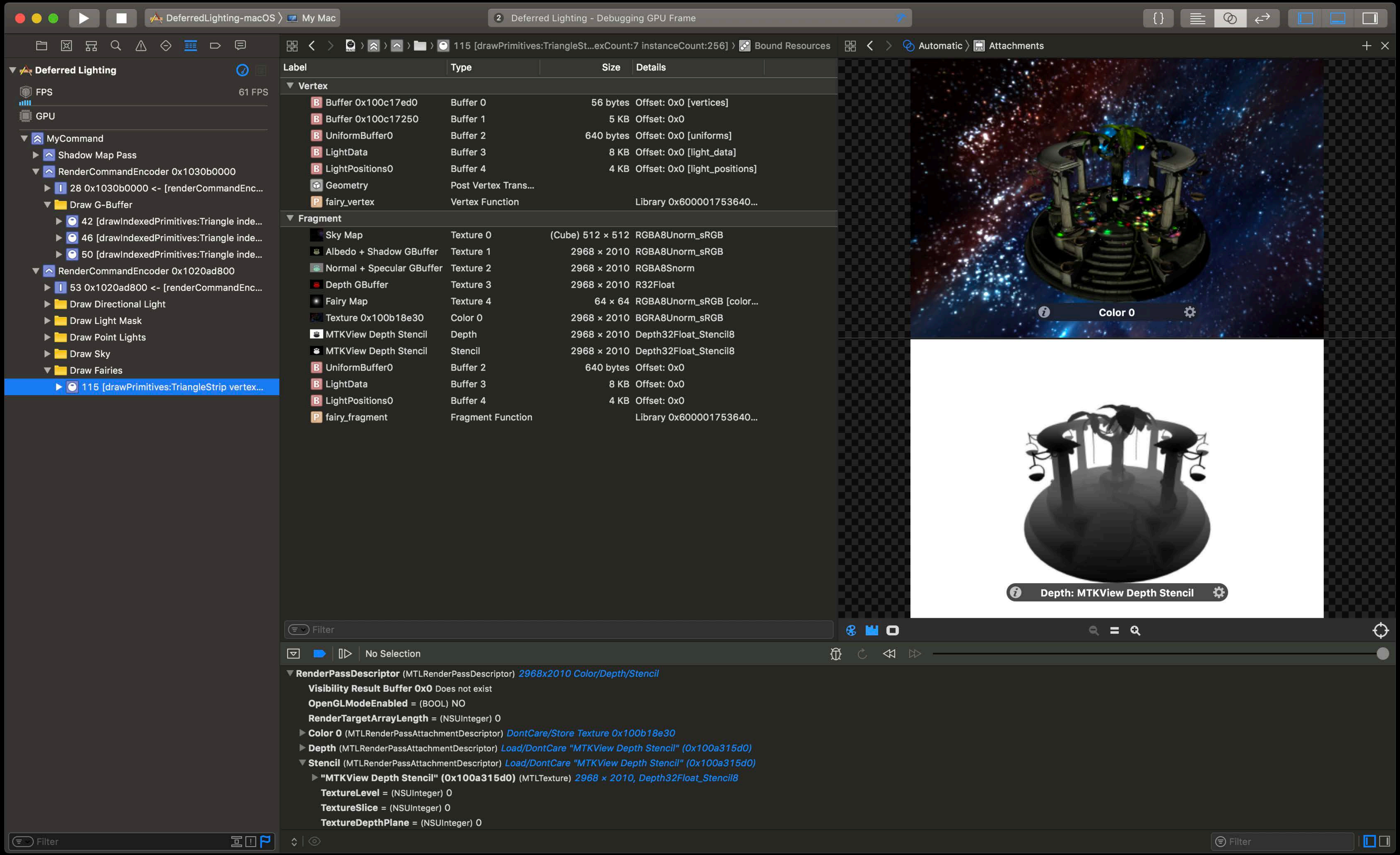
Integrated into Metal Frame Debugger



Demo

Xavier Verguin Gonzalez

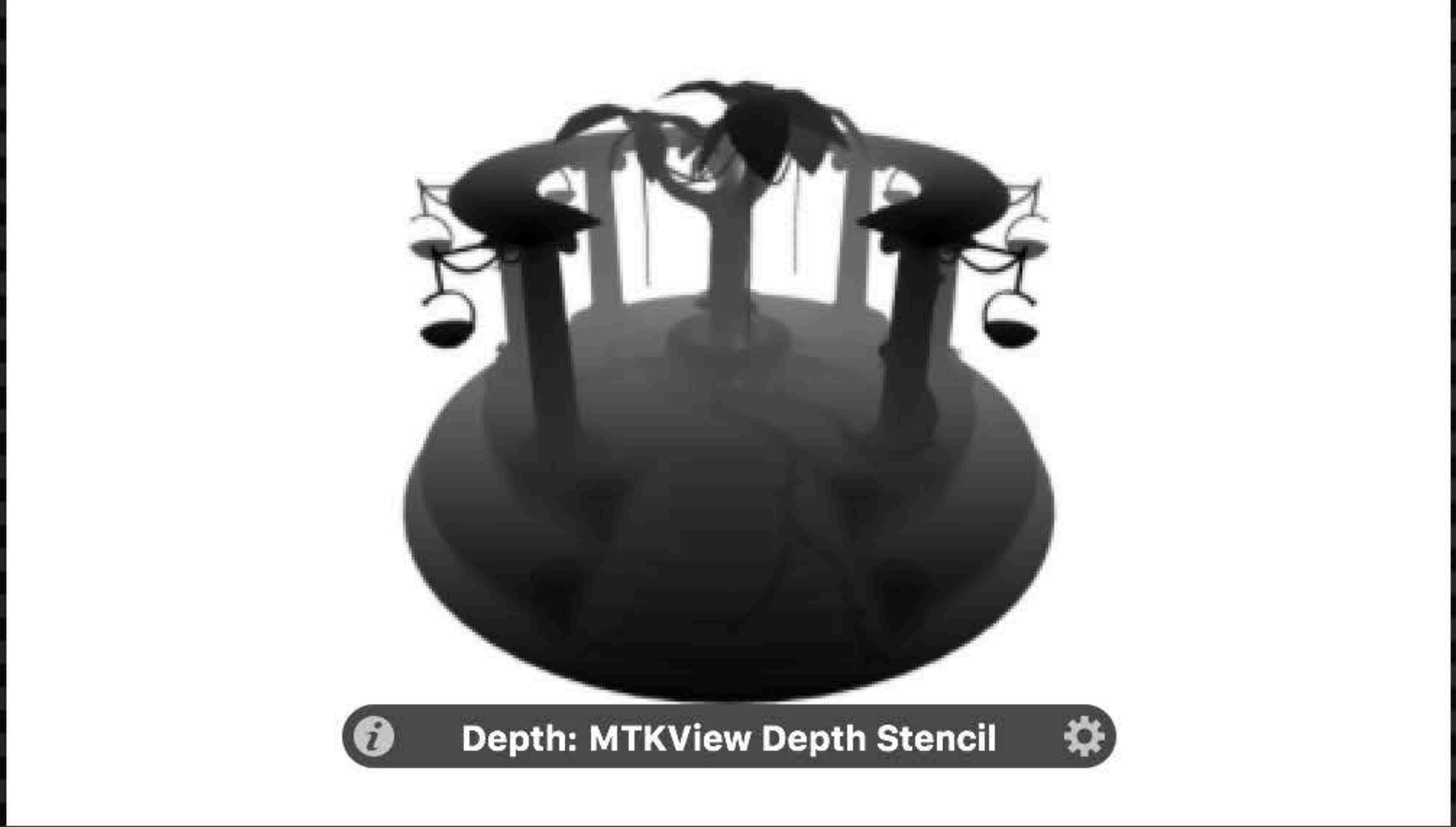
Starting the Shader Debugger



8 KB Offset: 0x0

4 KB Offset: 0x0

Library 0x600001753640...



/Depth/Stencil

ure 0x100b18e30

View Depth Stencil" (0x100a315d0)

View Depth Stencil" (0x100a315d0)

× 2010, Depth32Float_Stencil8


8 KB Offset: 0x0
4 KB Offset: 0x0
Library 0x600001753640...
Depth: MTKView Depth Stencil
/Depth/Stencil
ure 0x100b18e30
/iew Depth Stencil" (0x100a315d0)
View Depth Stencil" (0x100a315d0)
× 2010, Depth32Float_Stencil8


8 KB Offset: 0x0
4 KB Offset: 0x0
Library 0x600001753640...
/Depth/Stencil
ure 0x100b18e30
/iew Depth Stencil" (0x100a315d0)
View Depth Stencil" (0x100a315d0)
× 2010, Depth32Float_Stencil8
Vertex
Fragment
211 221 216 255
Color 0
1,630
987
Debug


8 KB Offset: 0x0
4 KB Offset: 0x0
Library 0x600001753640...
/Depth/Stencil
ure 0x100b18e30
View Depth Stencil" (0x100a315d0)
View Depth Stencil" (0x100a315d0)
× 2010, Depth32Float_Stencil8
Vertex
Fragment
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
0
0
0
Threads per Grid
480
368
1
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

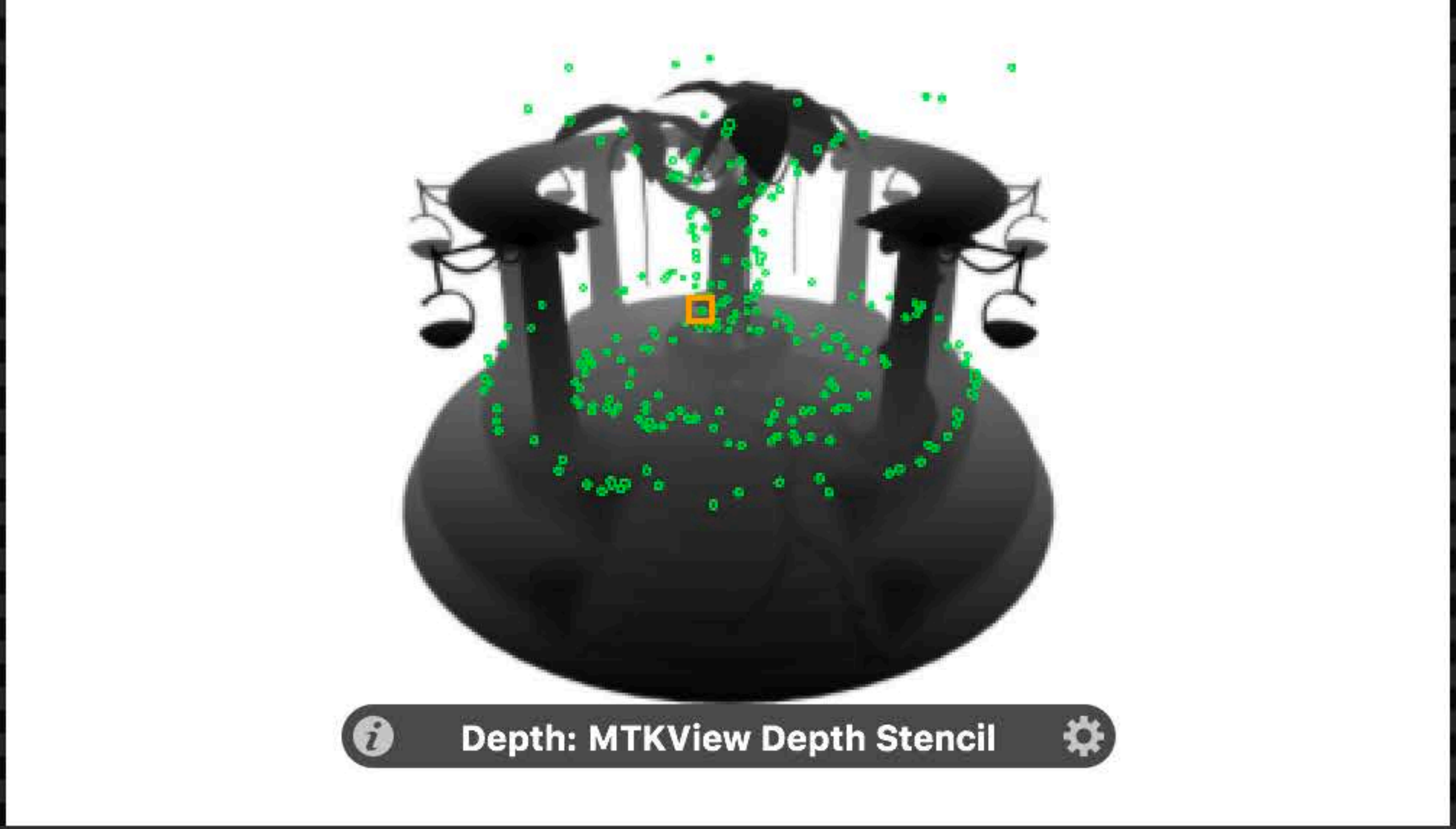

X
Y
Z
Thread Position in Grid
0
0
0
Threads per Grid
480
368
1
Thread Position in Threadgroup
0
0
0
Threads per Threadgroup
16
16
1
Threadgroup Position in Grid
0
0
0
Threadgroups per Grid
30
23
1
Debug

pler;

c = [0.037841797, 0.03

fragColor = [0.23693848

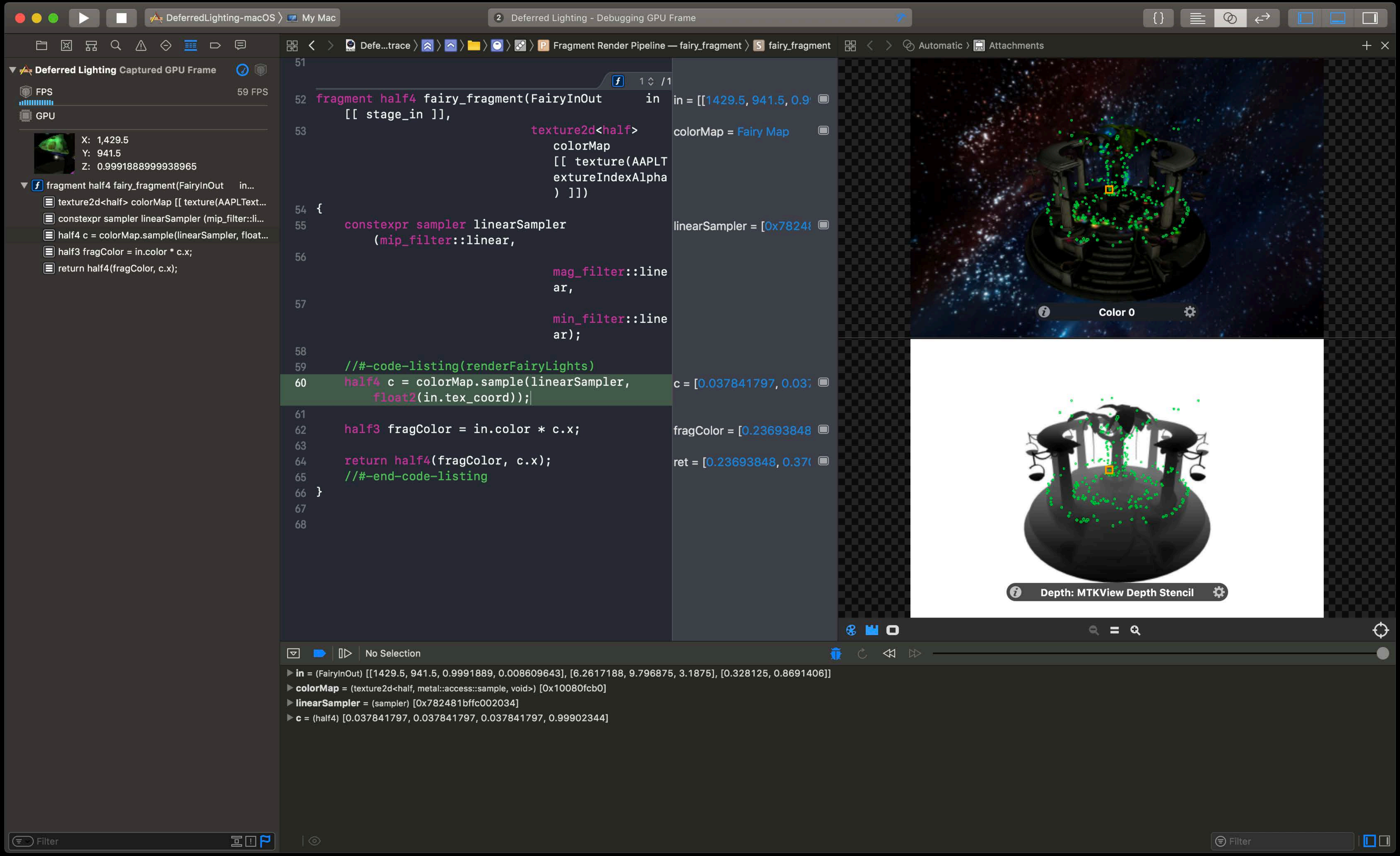
ret = [0.23693848, 0.37(



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

0]

4]


DeferredLighting-macOS
My Mac
Deferred Lighting - Debugging GPU Frame
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);
Defe...trace
Fragment Render Pipeline — fairy_fragment
fairy_fragment
Automatic
Attachments
fragment half4 fairy_fragment(FairyInOut in [[stage_in]],
texture2d<half> colorMap [[texture(AAPLTextureIndexAlpha)]])
{
constexpr sampler linearSampler (mip_filter::linear,
mag_filter::linear,
min_filter::linear);
//#-code-listing(renderFairyLights)
half4 c = colorMap.sample(linearSampler, float2(in.tex_coord));
half3 fragColor = in.color * c.x;
return half4(fragColor, c.x);
//#-end-code-listing
}
in = [[1429.5, 941.5, 0.9
colorMap = Fairy Map
linearSampler = [0x7824
c = [0.037841797, 0.03
fragColor = [0.23693848
ret = [0.23693848, 0.37
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]
Filter
Filter

Inspecting Variables

Just go the source line

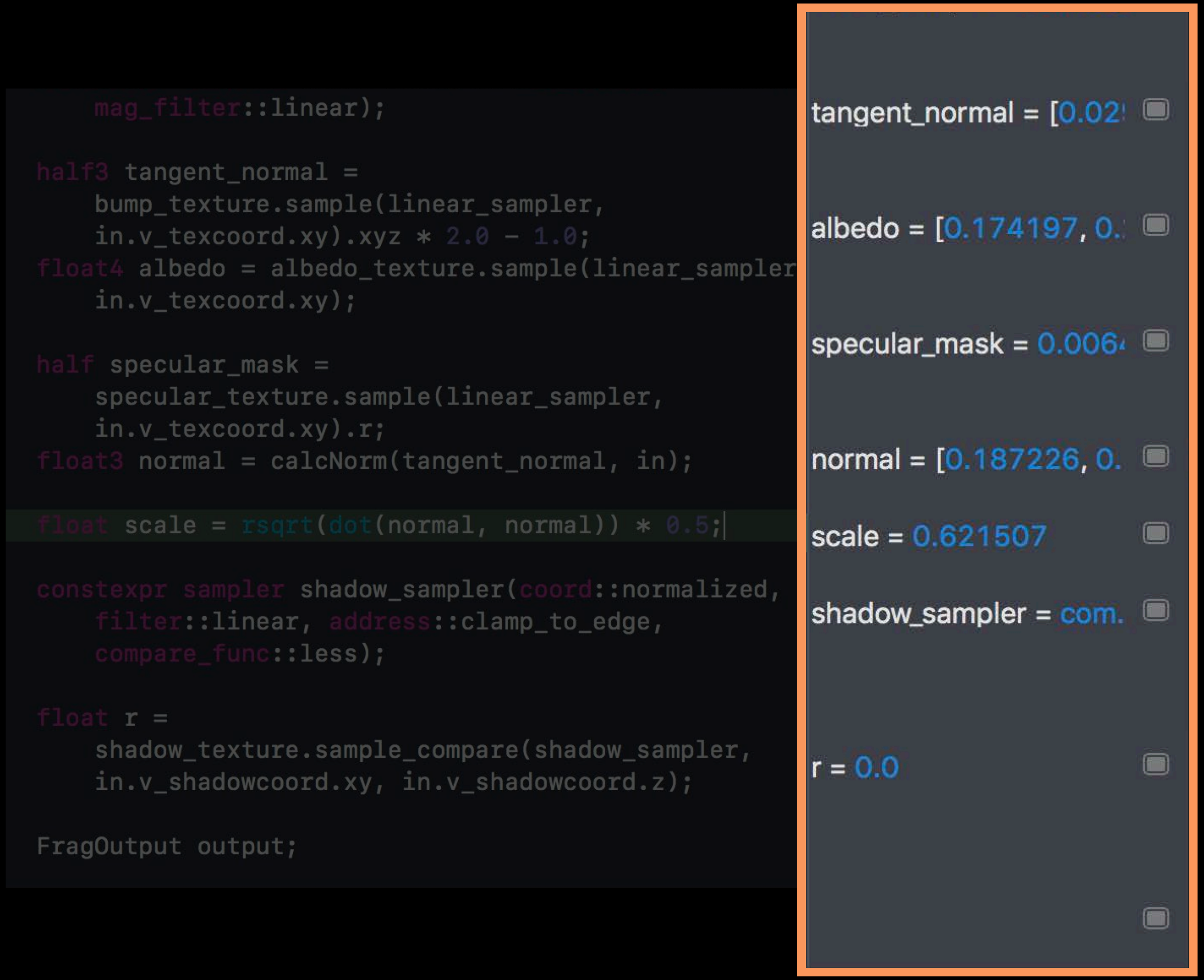
```
    mag_filter::linear);

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 = [0.02
albedo = [0.174197, 0.
specular_mask = 0.006
normal = [0.187226, 0.
scale = 0.621507
shadow_sampler = com.
r = 0.0

Inspecting Variables

Just go the source line

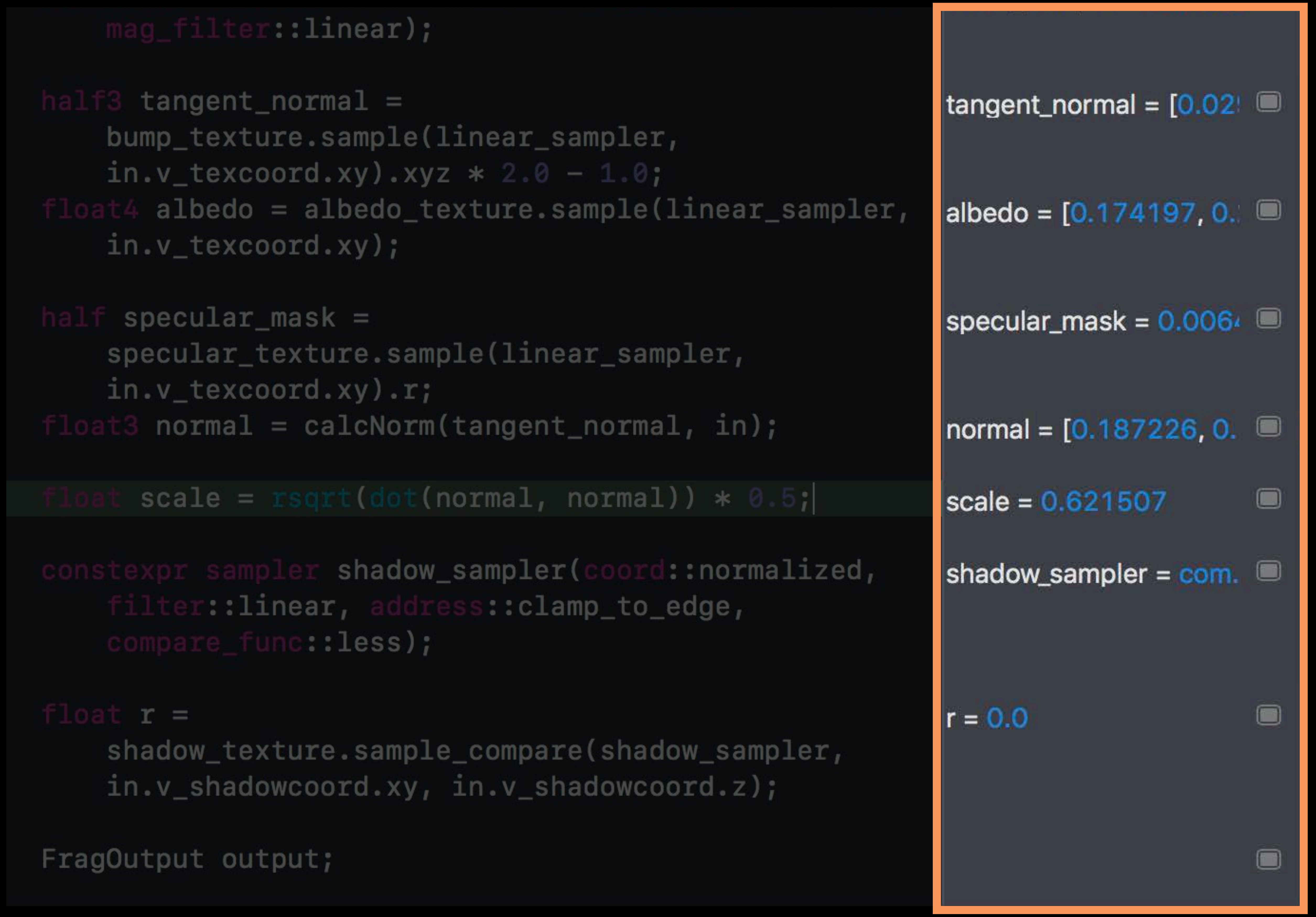
- The side bar for the modified variables



Inspecting Variables

Just go the source line

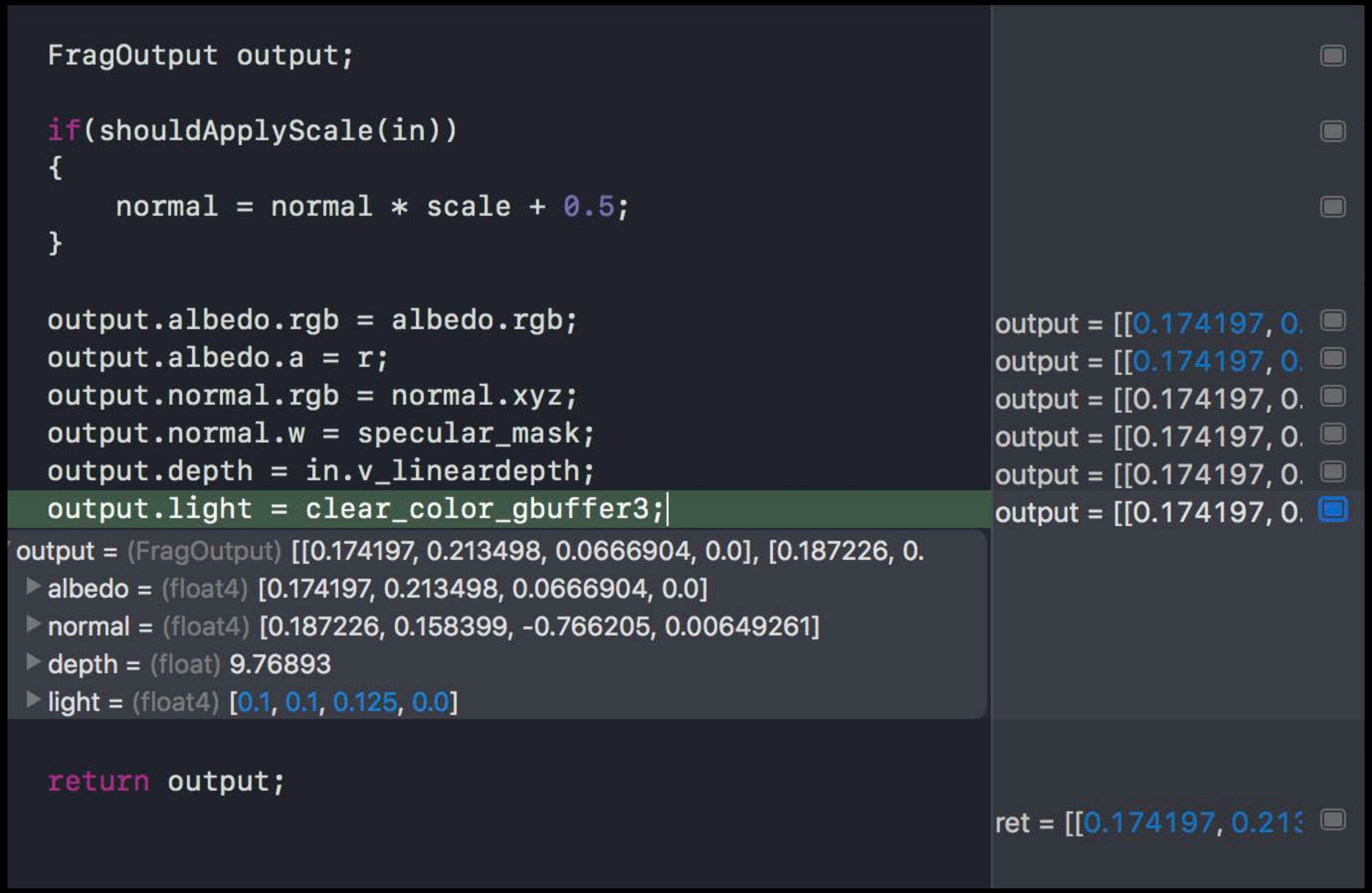
- The side bar for the modified variables



Inspecting Variables

Just go the source line

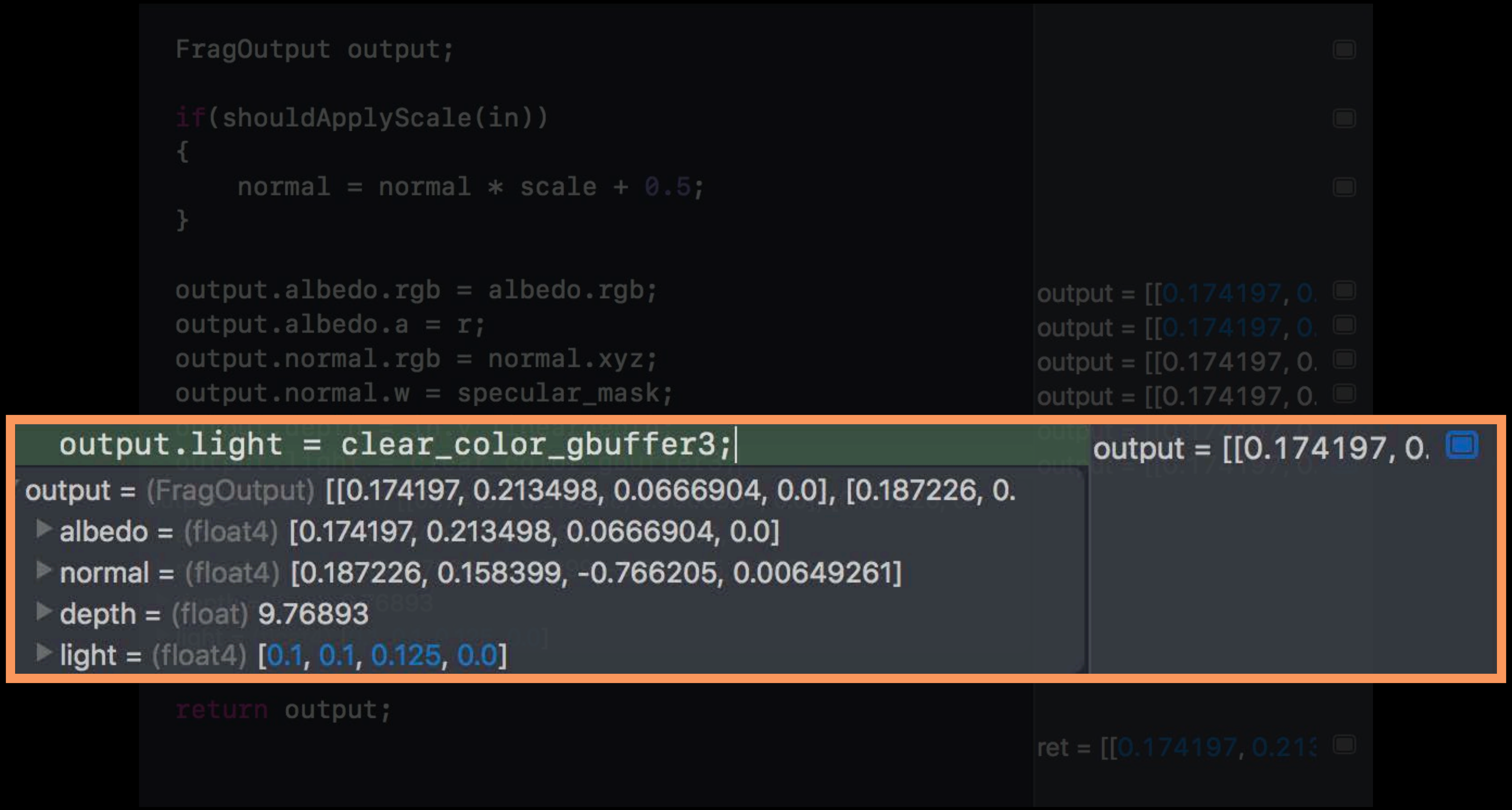
- The side bar for the modified variables
- Details view for full information



Inspecting Variables

Just go the source line

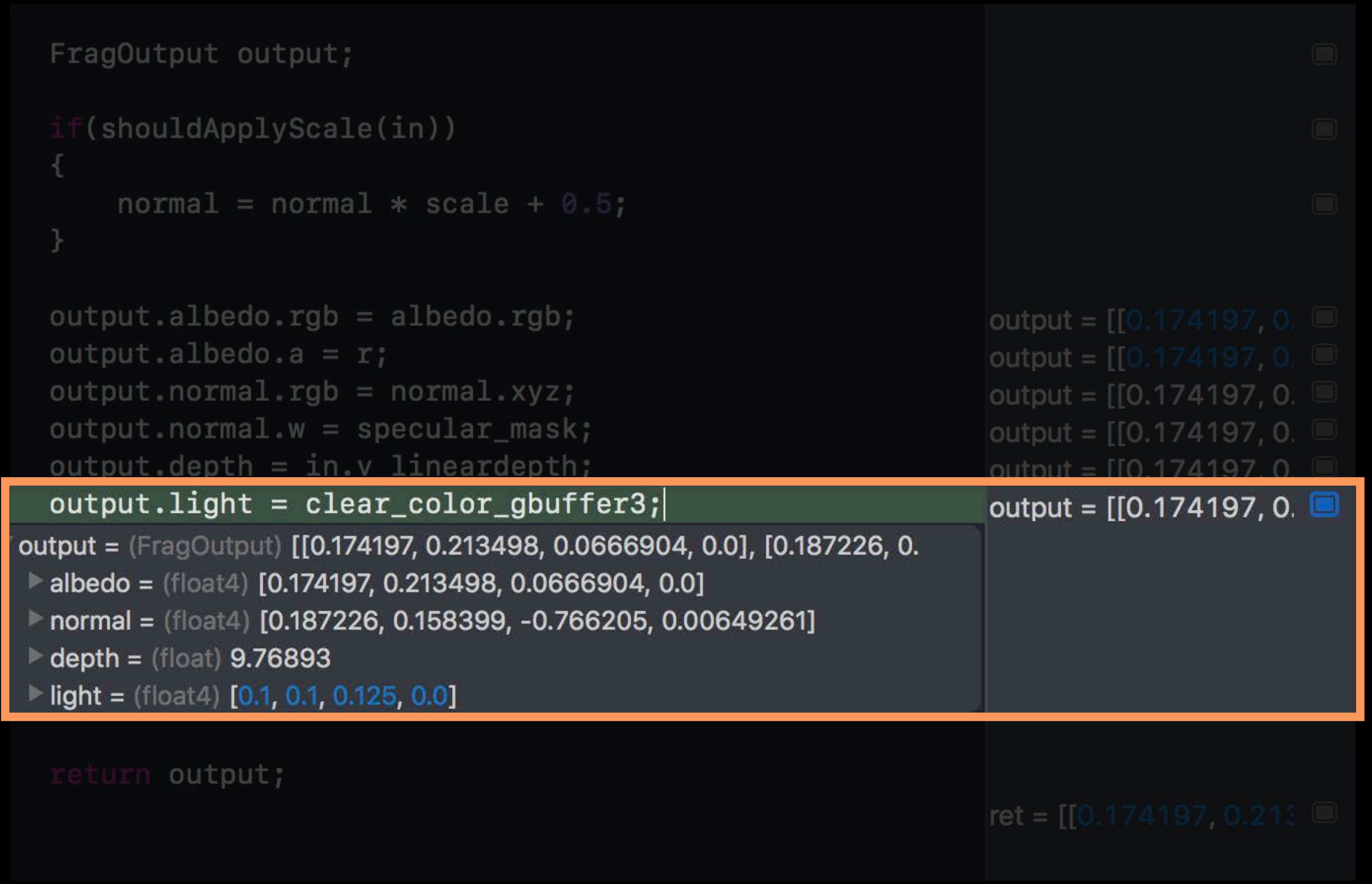
- The side bar for the modified variables
- Details view for full information



Inspecting Variables

Just go the source line

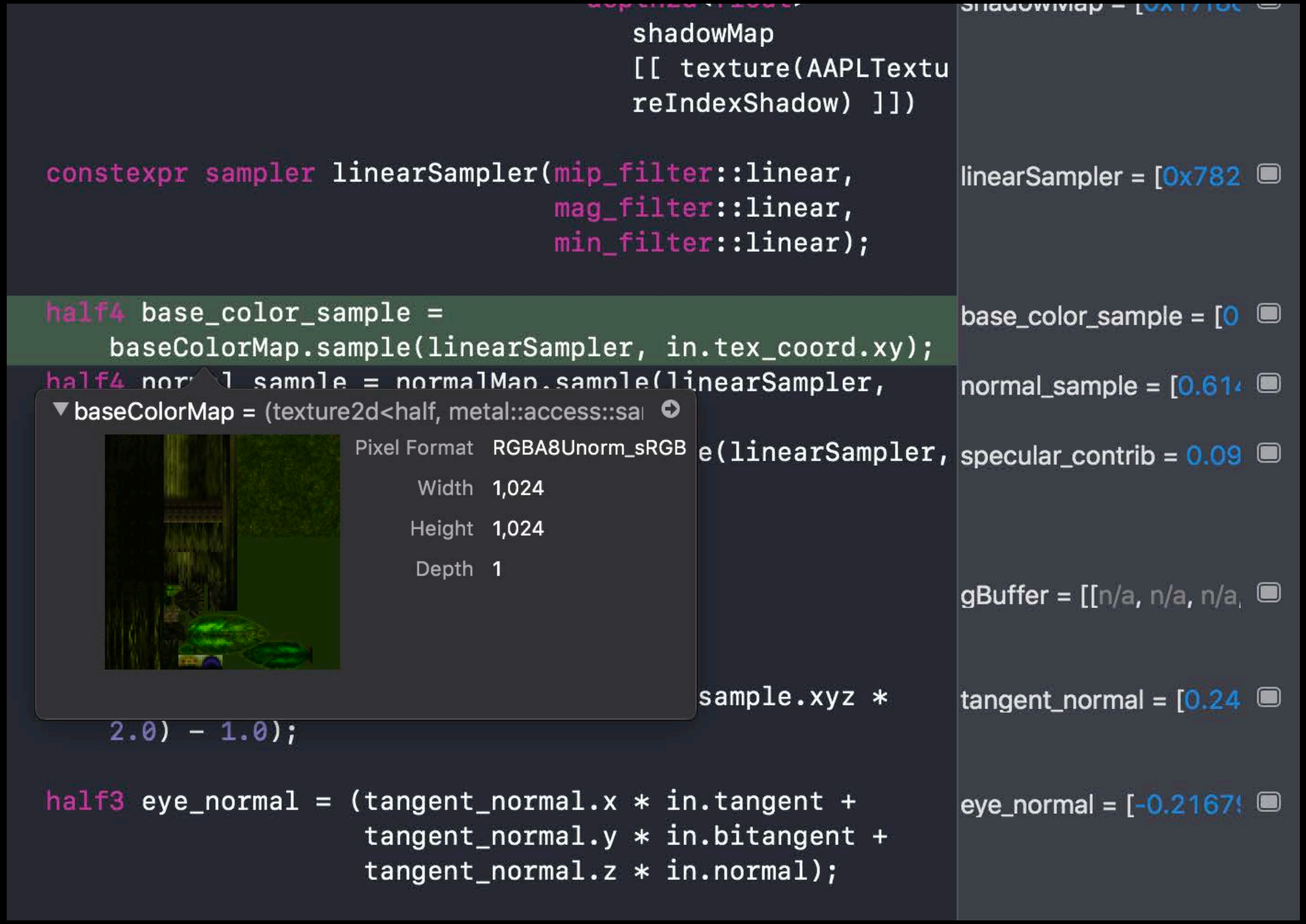
- The side bar for the modified variables
- Details view for full information



Inspecting Variables

Just go the source line

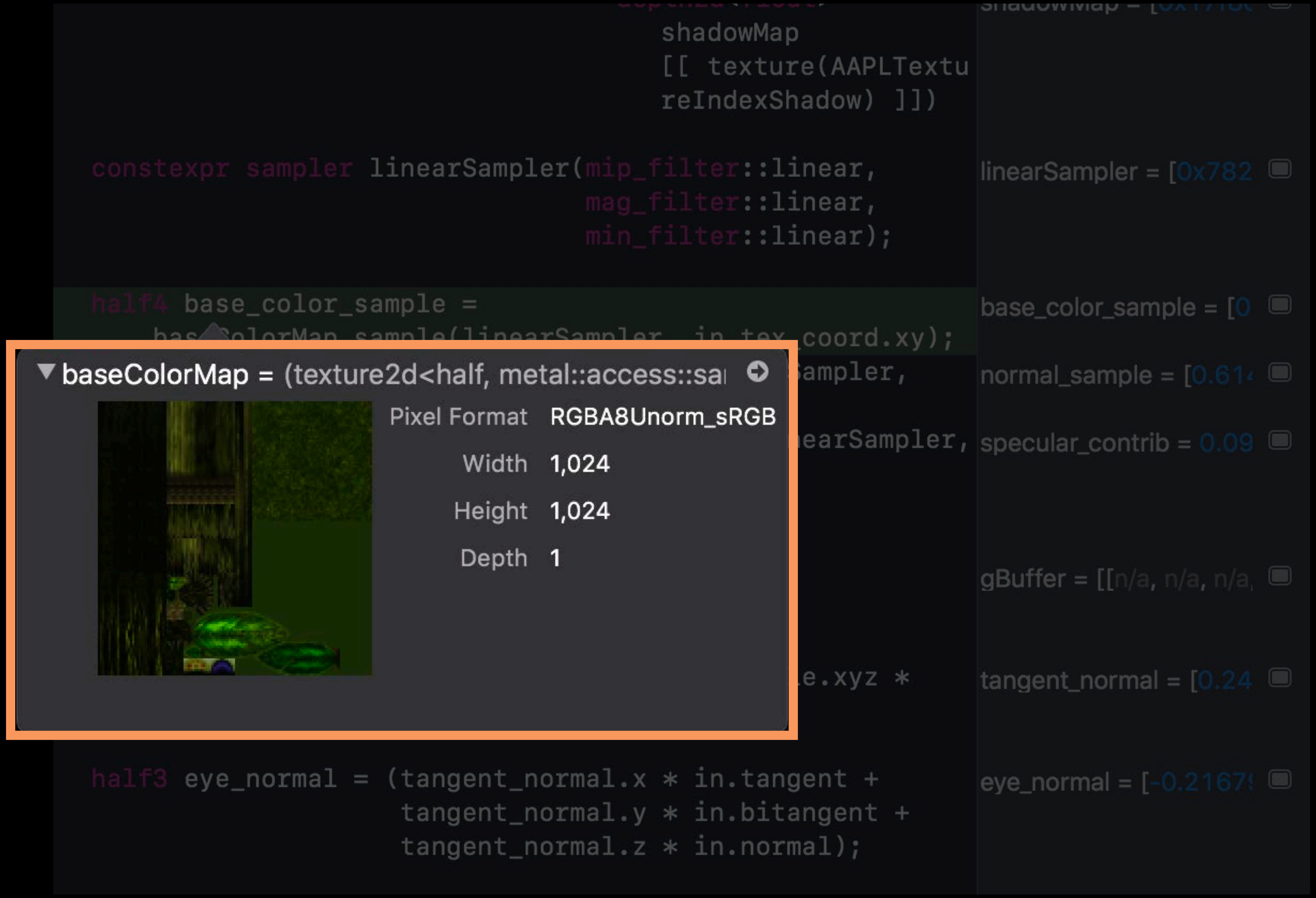
- The side bar for the modified variables
- Details view for full information
- Hover for in place access



Inspecting Variables

Just go the source line

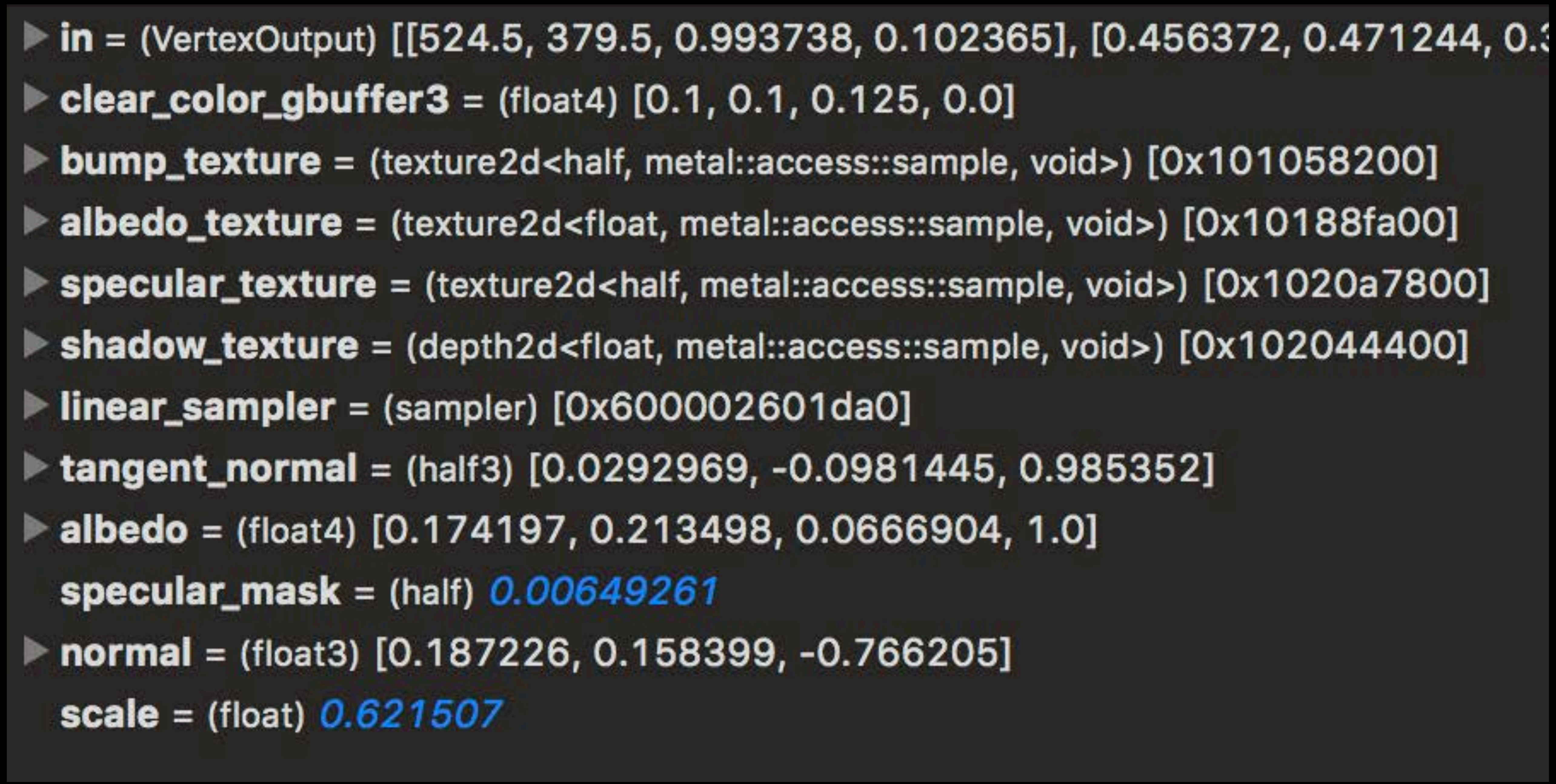
- The side bar for the modified variables
- Details view for full information
- Hover for in place access



Inspecting Variables

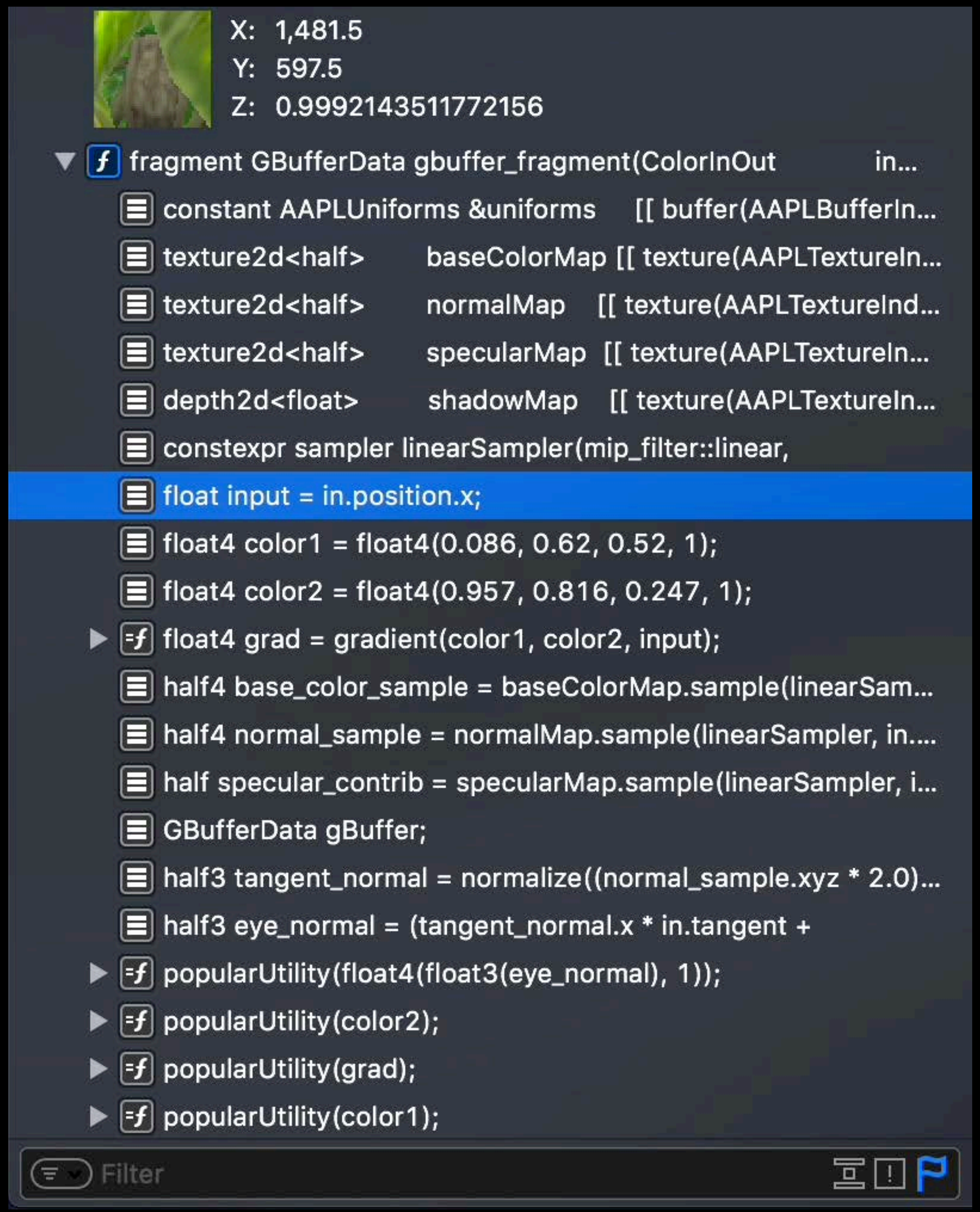
Just go the source line

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



Following the Execution

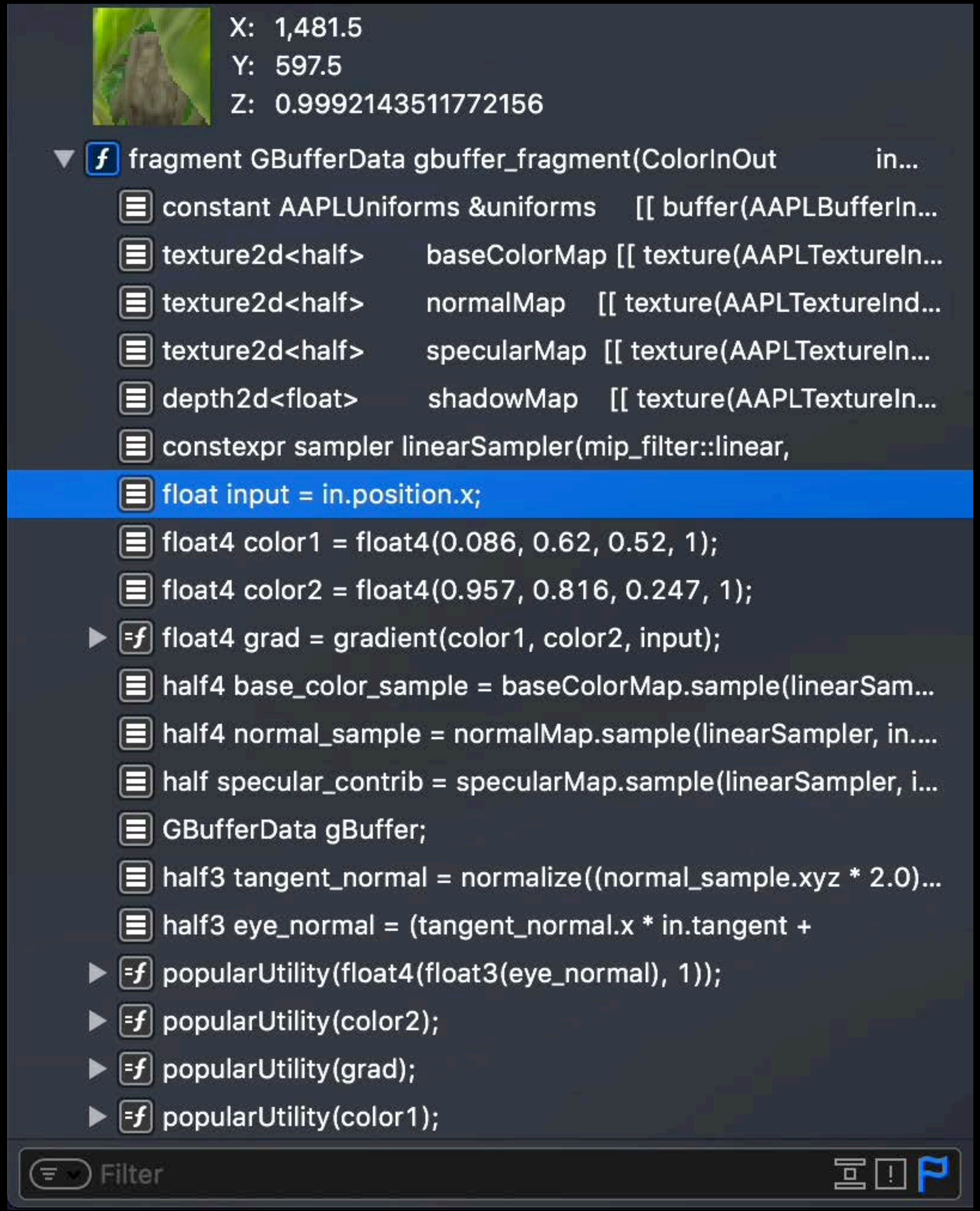
All source lines executed are available in navigator



Following the Execution

All source lines executed are available in navigator

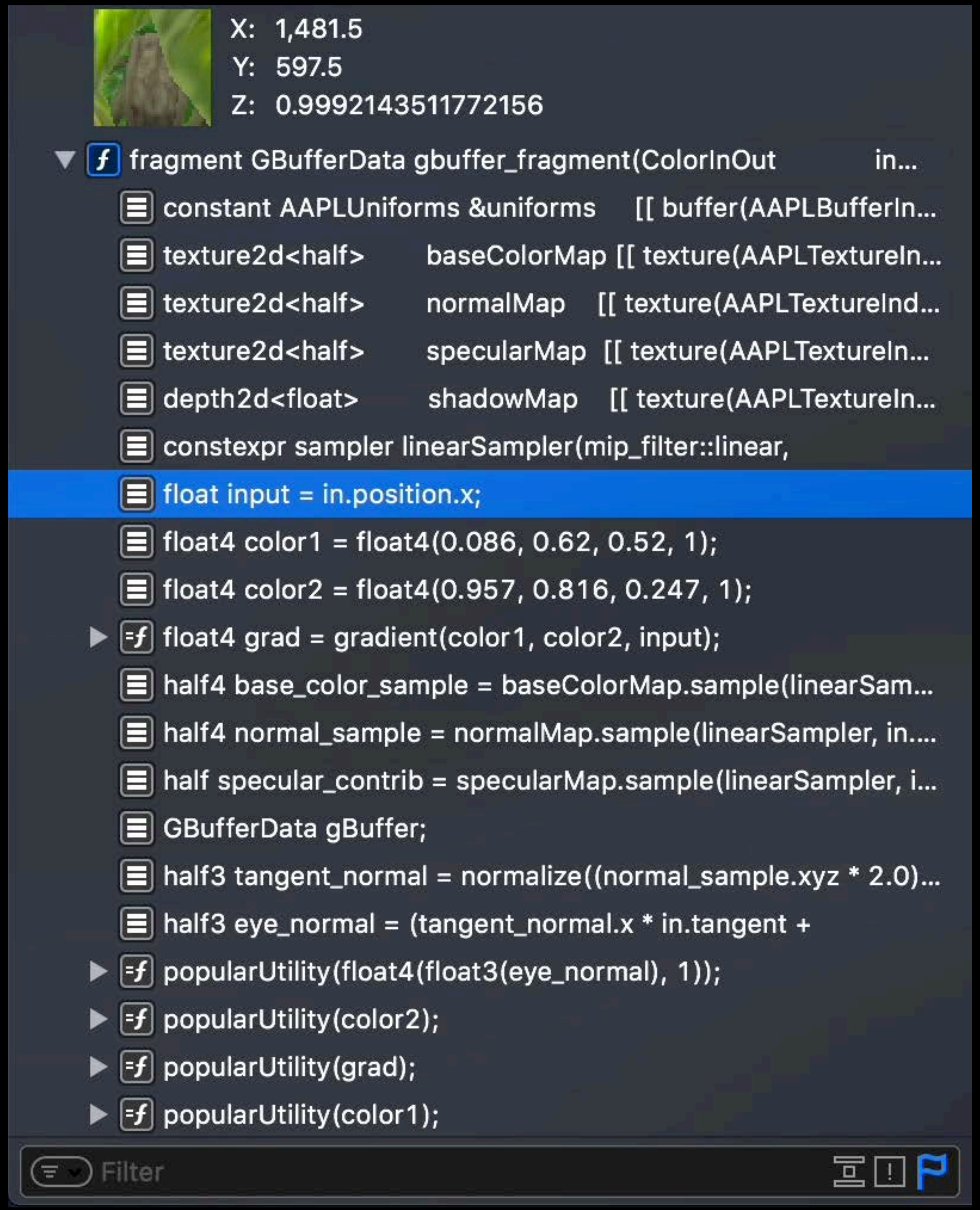
- Flexible stepping



Following the Execution

All source lines executed are available in navigator

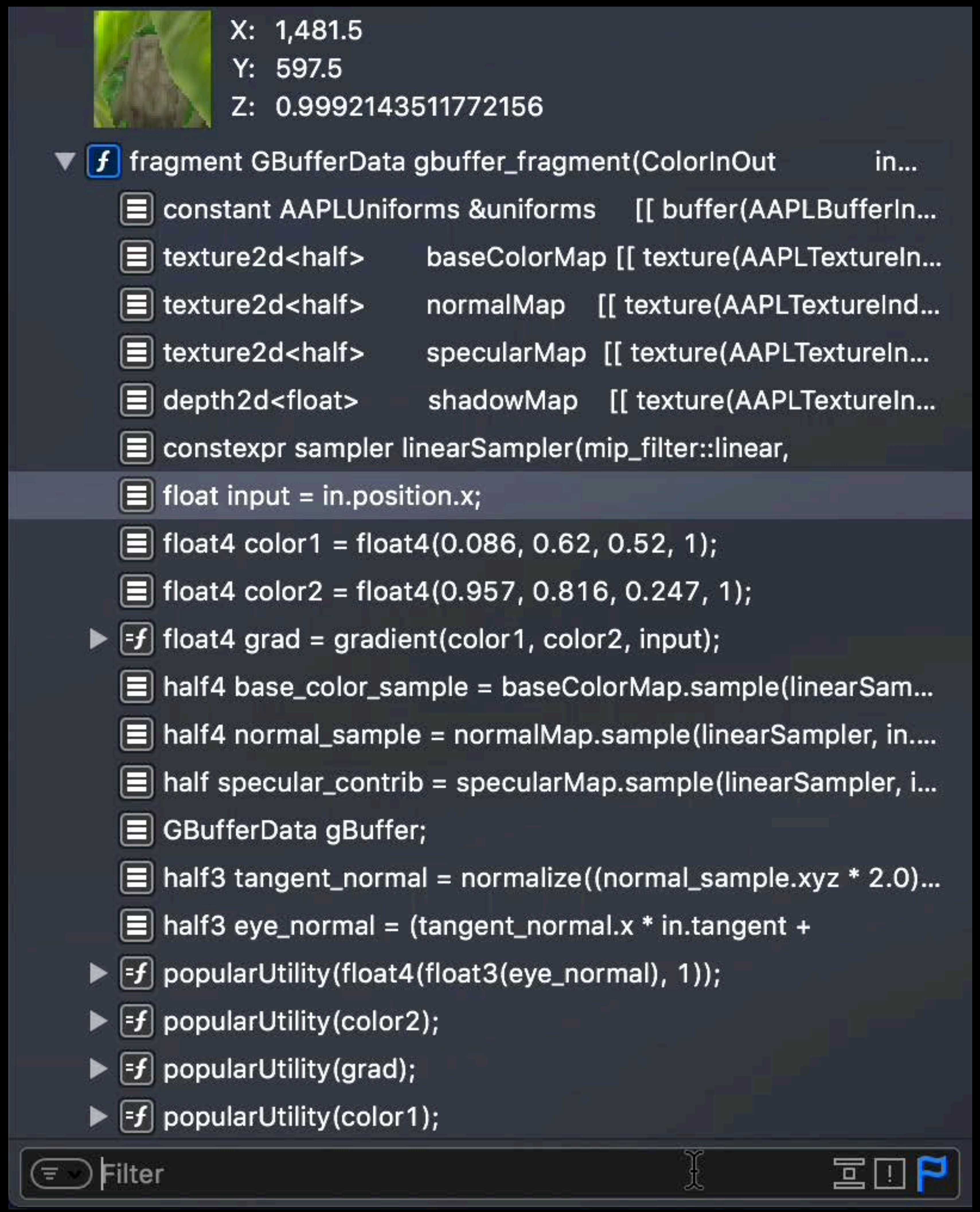
- Flexible stepping
  - Backwards debugging



Following the Execution

All source lines executed are available in navigator

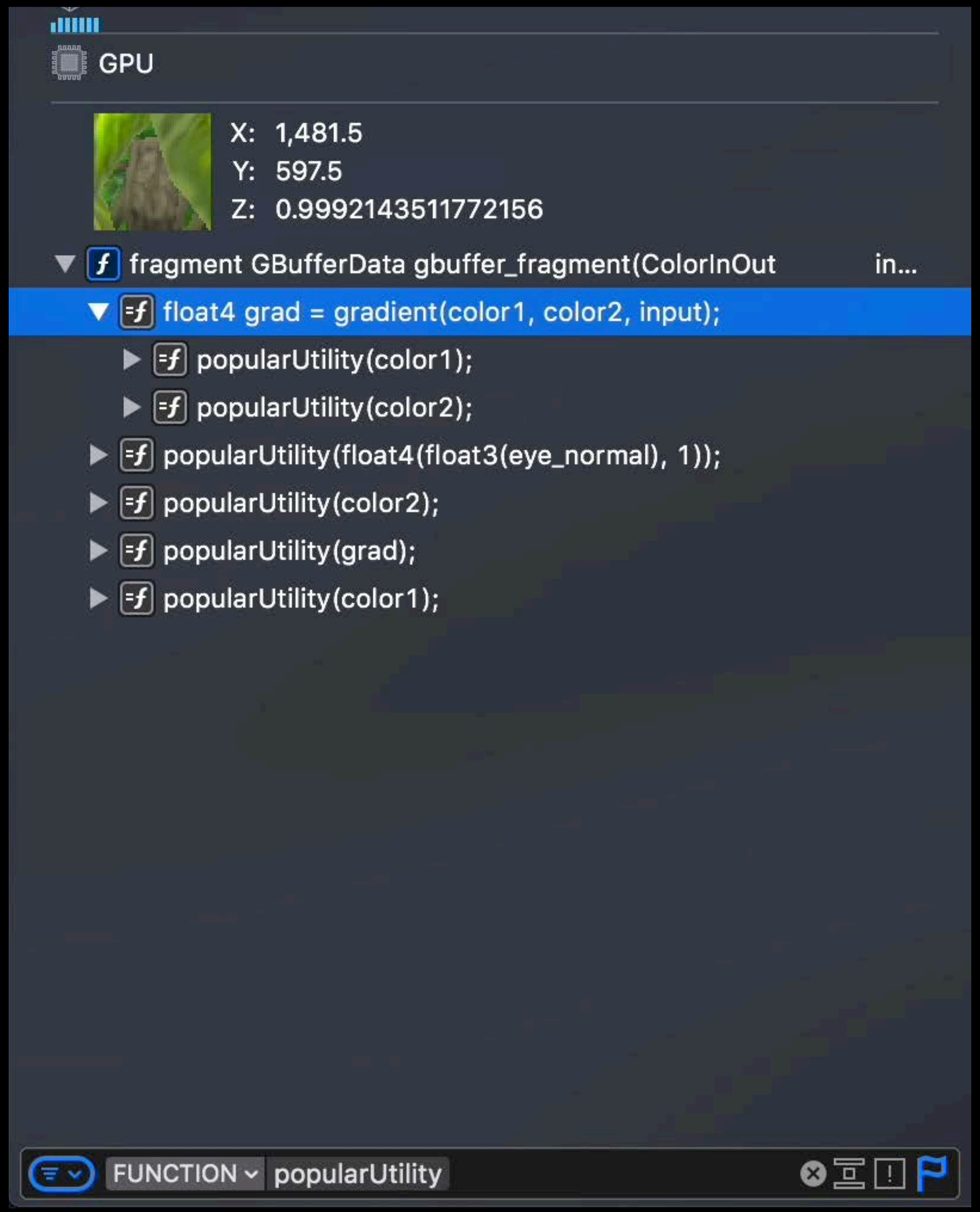
- Flexible stepping
  - Backwards debugging
- Use filter to focus



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);

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 =
```

```
grad = [0.76177585, 0.772069,
```

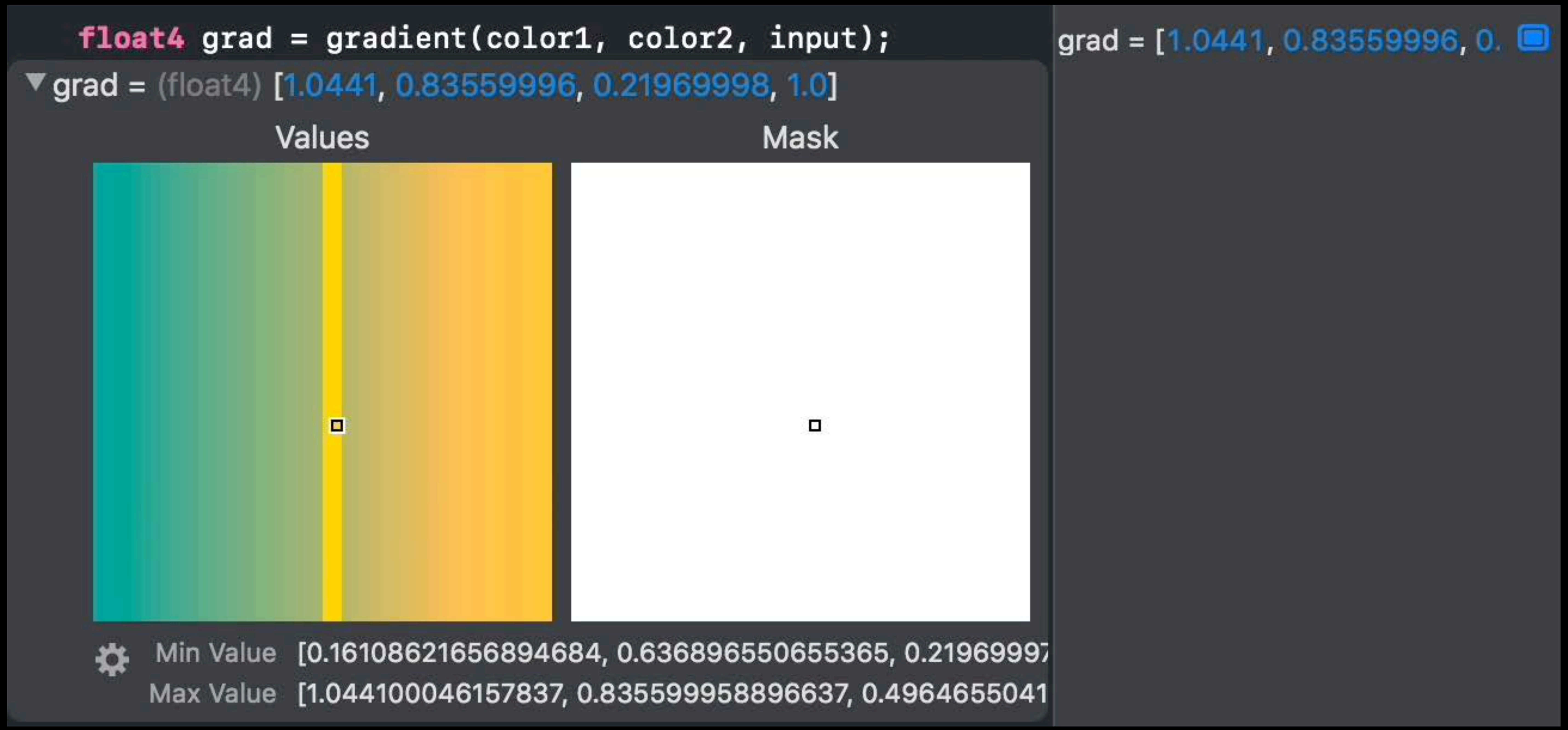
See Variables in Context

```
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)
```

See Variables in Context

Helps you understand your code

Important to for comparing good/bad values

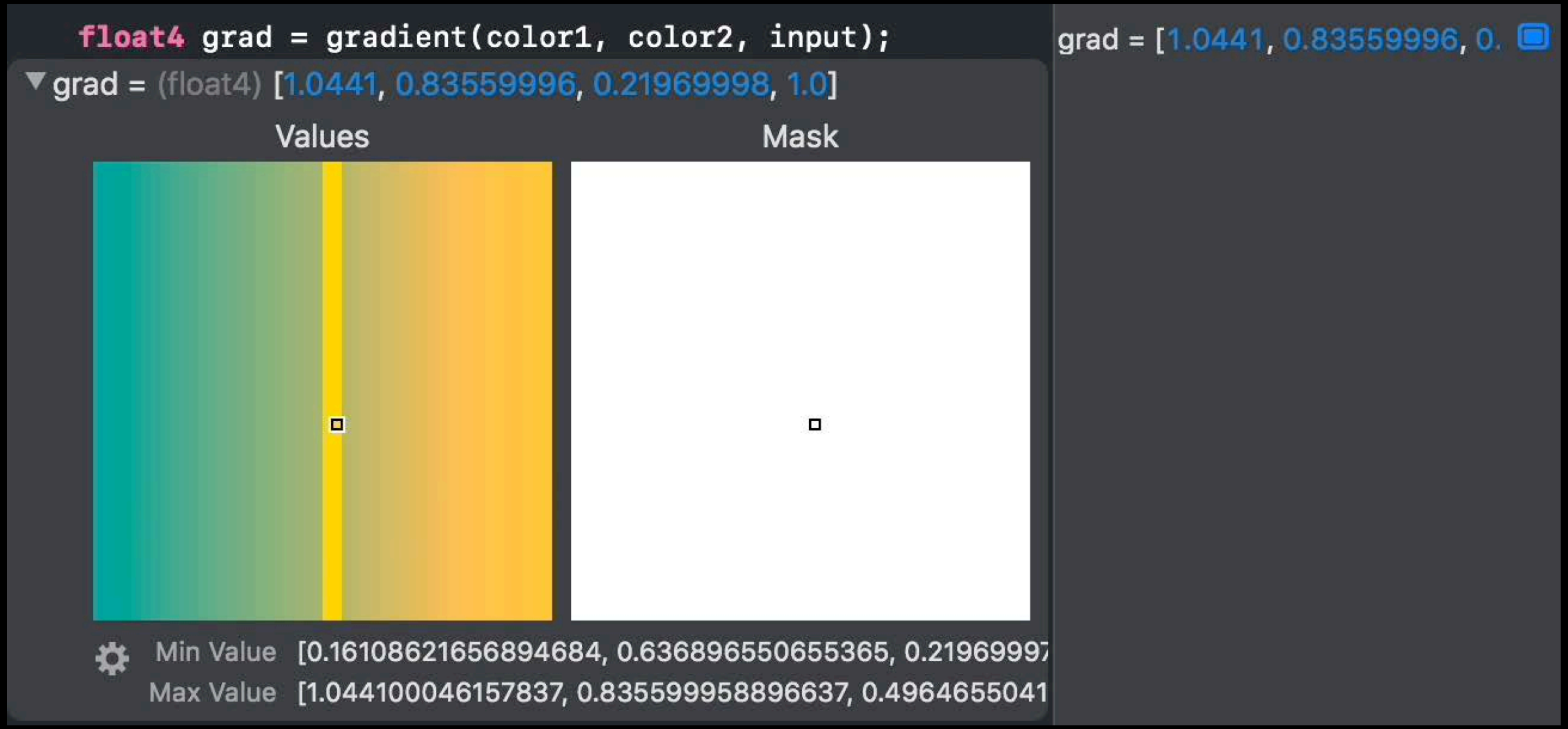


See Variables in Context

Helps you understand your code

Important to for comparing good/bad values

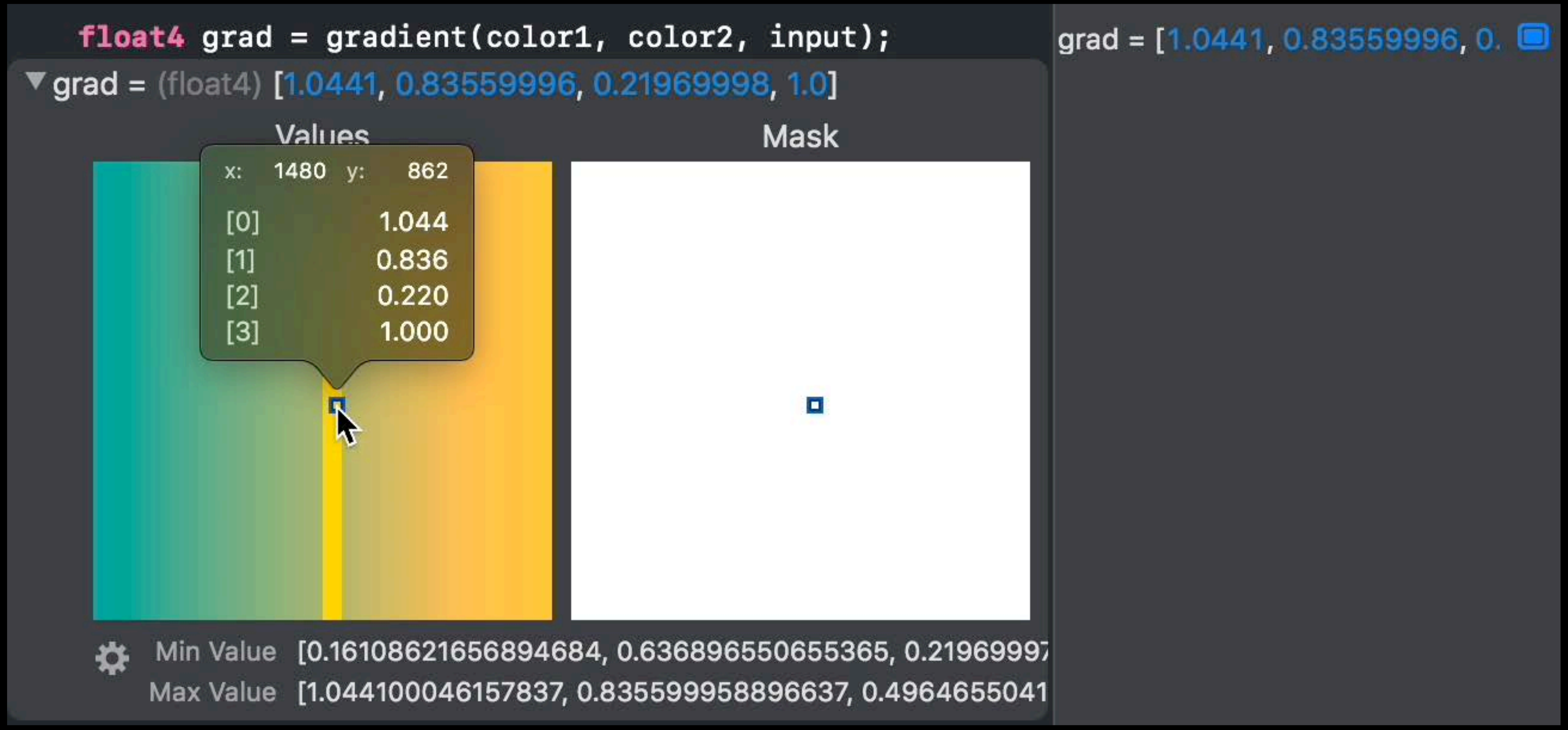
Hover for instant access



Comparing Good and Bad Pixels

Quickly compare threads

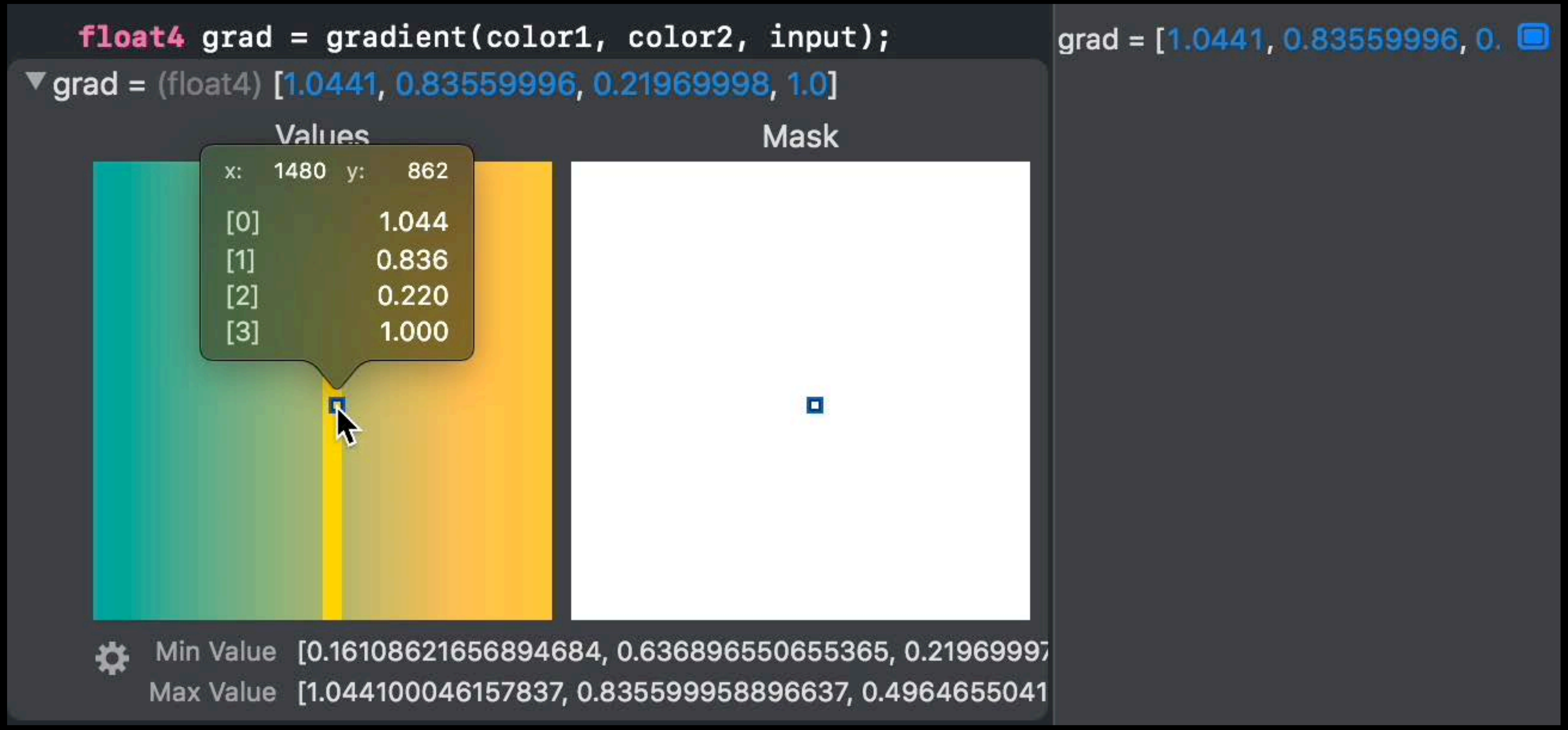
Execution history and variables are updated for the selected thread



Comparing Good and Bad Pixels

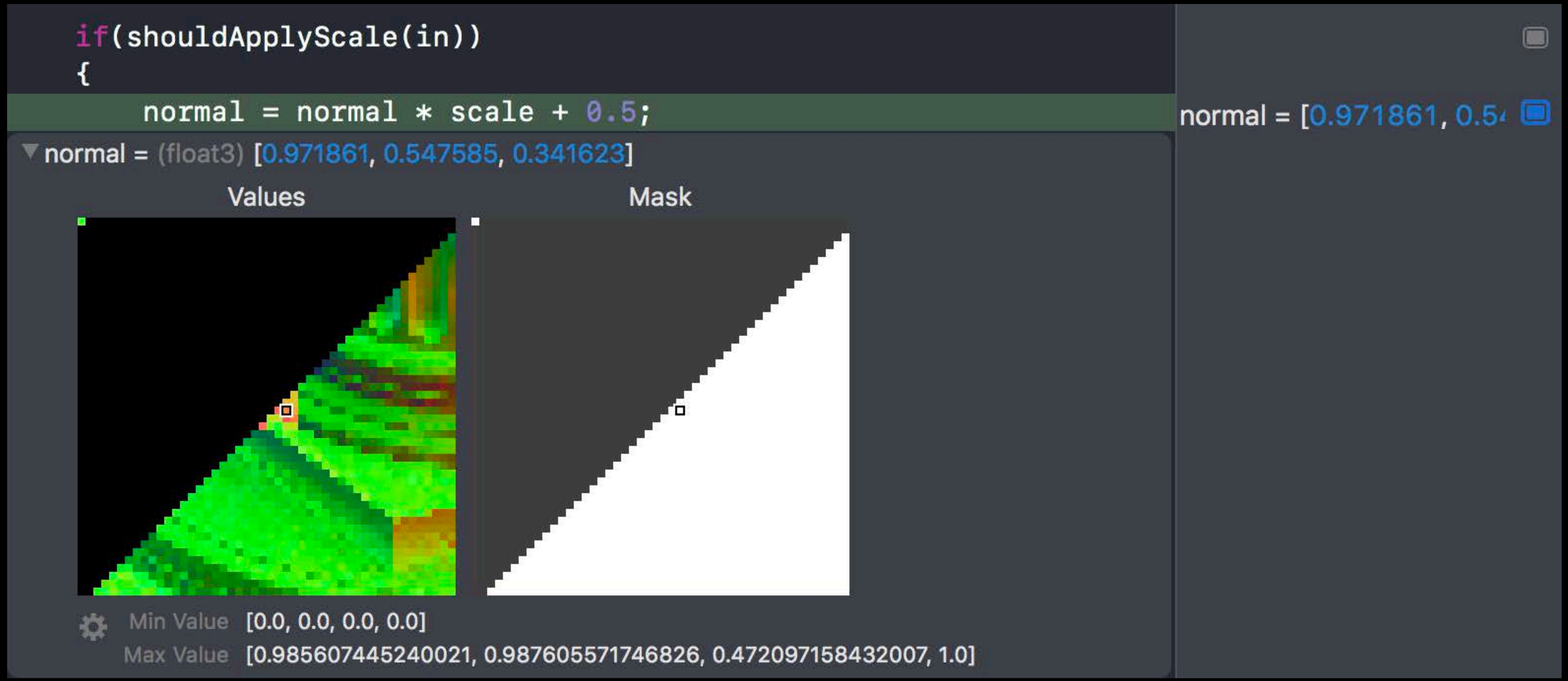
Quickly compare threads

Execution history and variables are updated for the selected thread



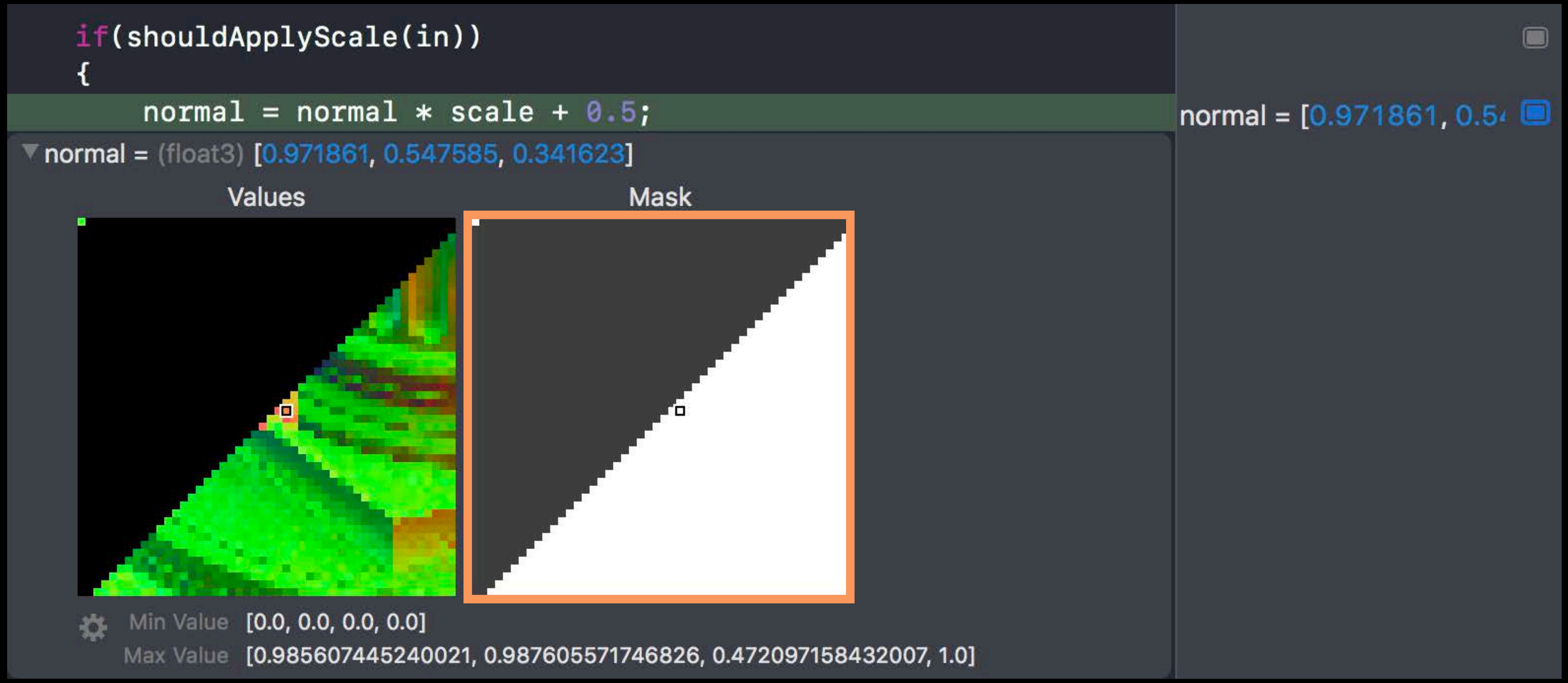
Understanding Divergence

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

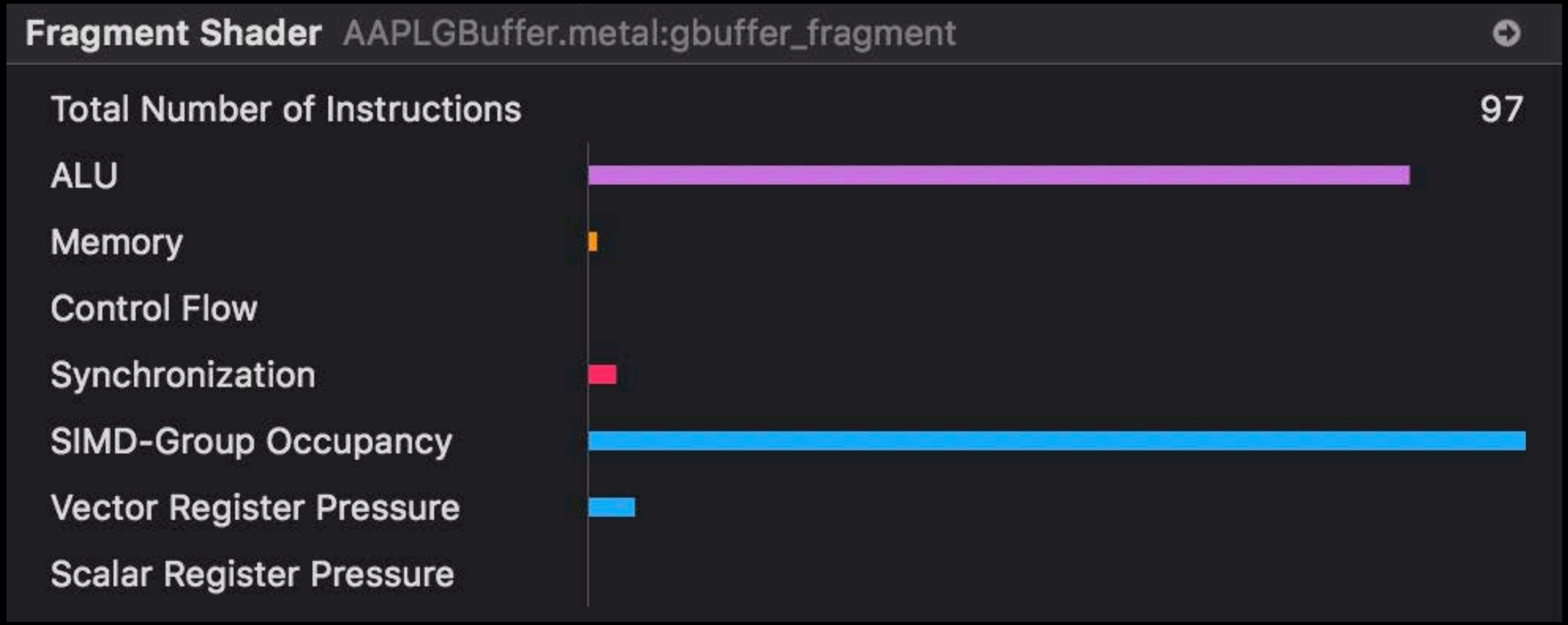
Profiling tools built into Metal Frame Debugger

- GPU counters

Knowing What to Optimize

Profiling tools built into Metal Frame Debugger

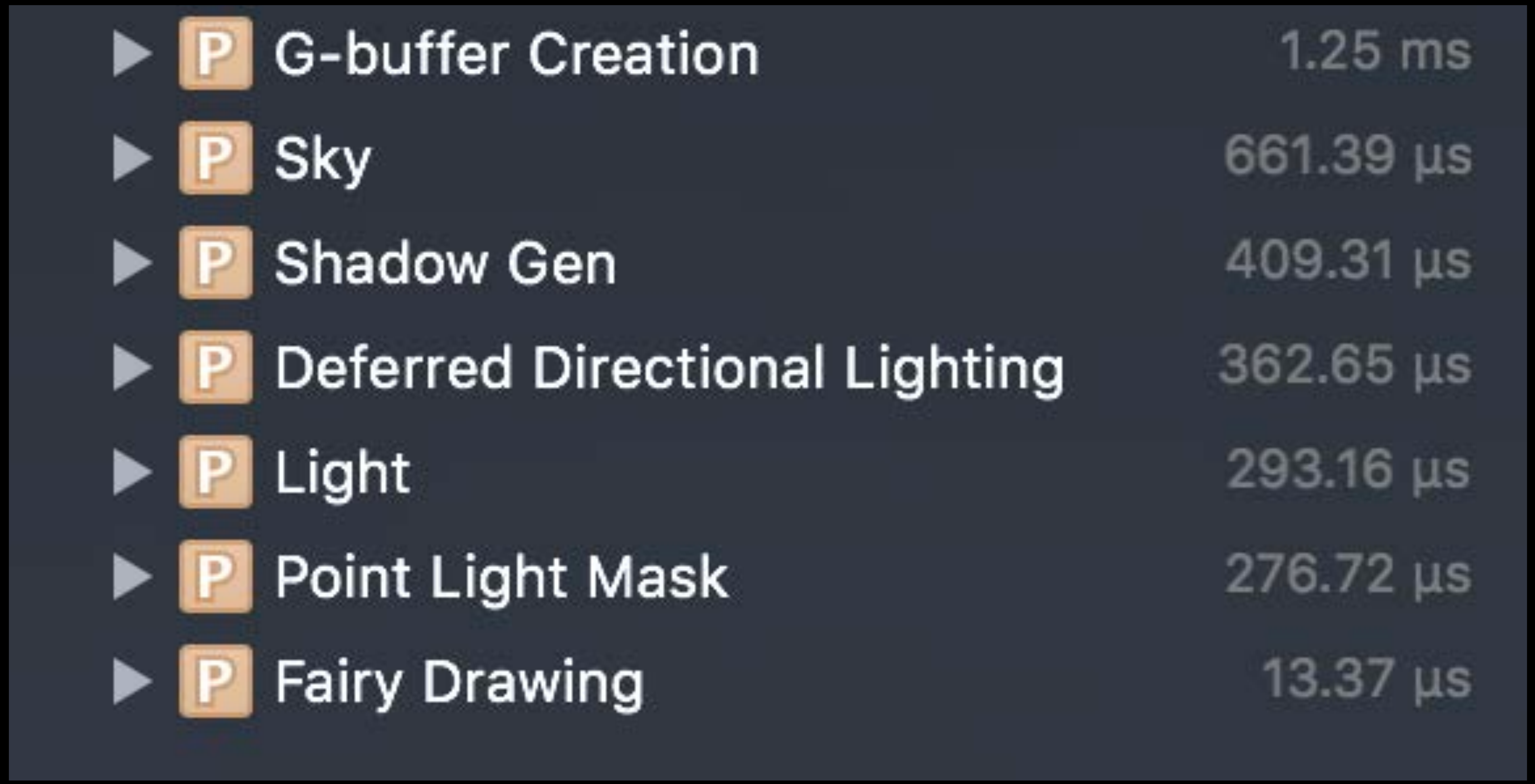
- GPU counters
- Pipeline statistics



Knowing What to Optimize

Profiling tools built into
Metal Frame Debugger

- GPU counters
- Pipeline statistics
- Shader profiler

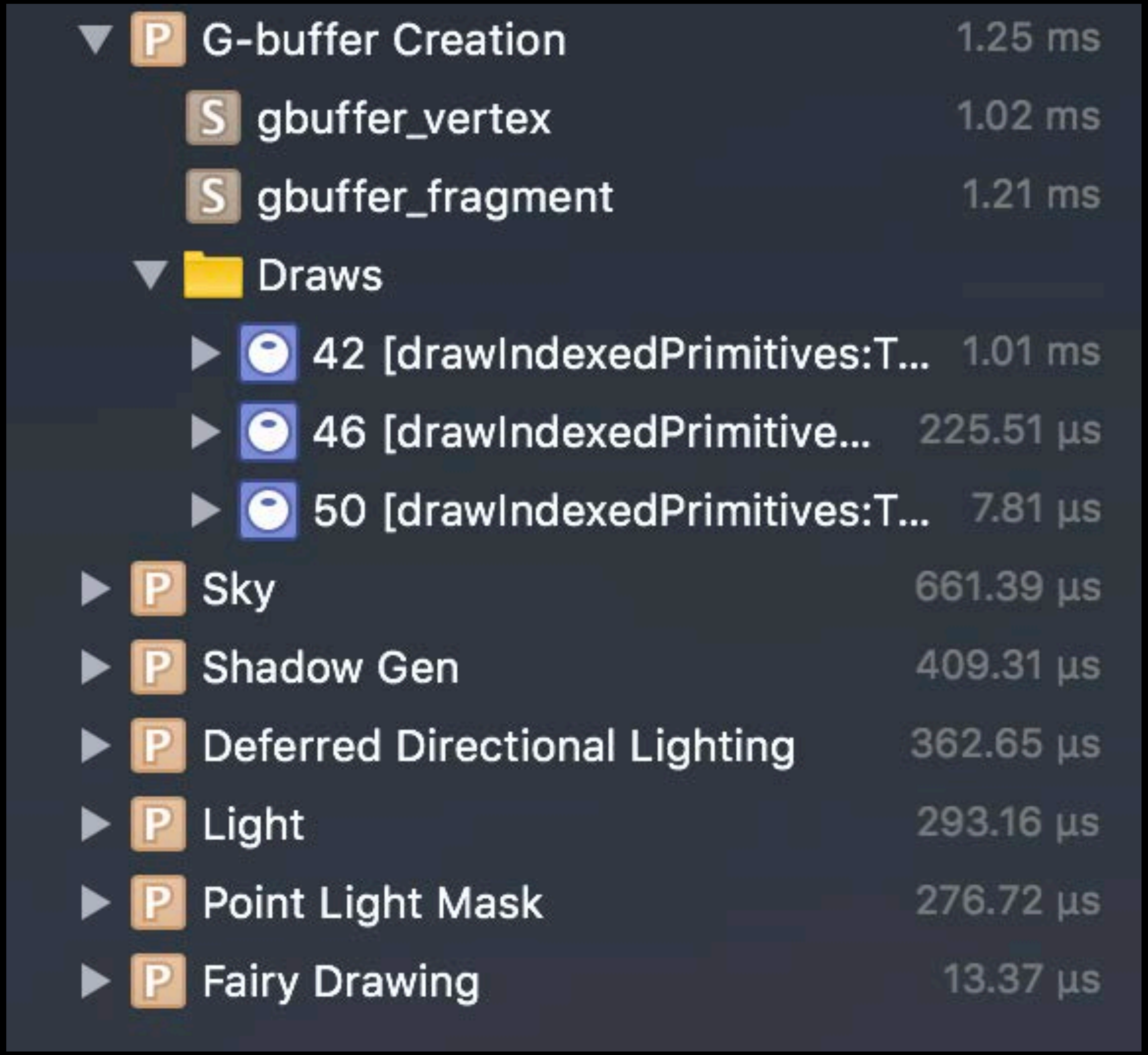


Shader Profiler

Provides per-pipeline timing information

Shader Profiler

Provides per-pipeline timing information



Shader Profiler

Provides per-pipeline timing information

Per-line execution cost (iOS and tvOS)

```
float3 colorSample = colorMap.sample(colorSampler,          45.71%
    in.texCoord.xy*0.15 - uniforms.time[0] *
    0.0075).xyz;

float3 colorSample2 = colorMap.sample(colorSampler,          1.53%
    in.texCoord.xy*0.05).xyz;

float relativeHeight =                                       0.43%
    clamp(in.positionModelSpace.y + 0.5, 0.0, 1.0);
float2 relativeXZ = clamp((in.positionModelSpace.xz         14.65%
    + 1.5) / 3.0, 0.0, 1.0); // Thi should be 1.0 /
    2.0

float spatialVariation = sin(3 *                             4.27%
    (in.positionModelSpace.x +
    in.positionModelSpace.z));
```

Shader Profiler

Provides per-pipeline timing information

Per-line execution cost (iOS and tvOS)

Shader edit and reload

```
float3 colorSample = colorMap.sample(colorSampler,          45.71%
    in.texCoord.xy*0.15 - uniforms.time[0] *
    0.0075).xyz;

float3 colorSample2 = colorMap.sample(colorSampler,          1.53%
    in.texCoord.xy*0.05).xyz;

float relativeHeight =                                       0.43%
    clamp(in.positionModelSpace.y + 0.5, 0.0, 1.0);
float2 relativeXZ = clamp((in.positionModelSpace.xz         14.65%
    + 1.5) / 3.0, 0.0, 1.0); // Thi should be 1.0 /
    2.0

float spatialVariation = sin(3 *                             4.27%
    (in.positionModelSpace.x +
    in.positionModelSpace.z));
```

Shader Profiler

Provides per-pipeline timing information

Per-line execution cost (iOS and tvOS)

Shader edit and reload

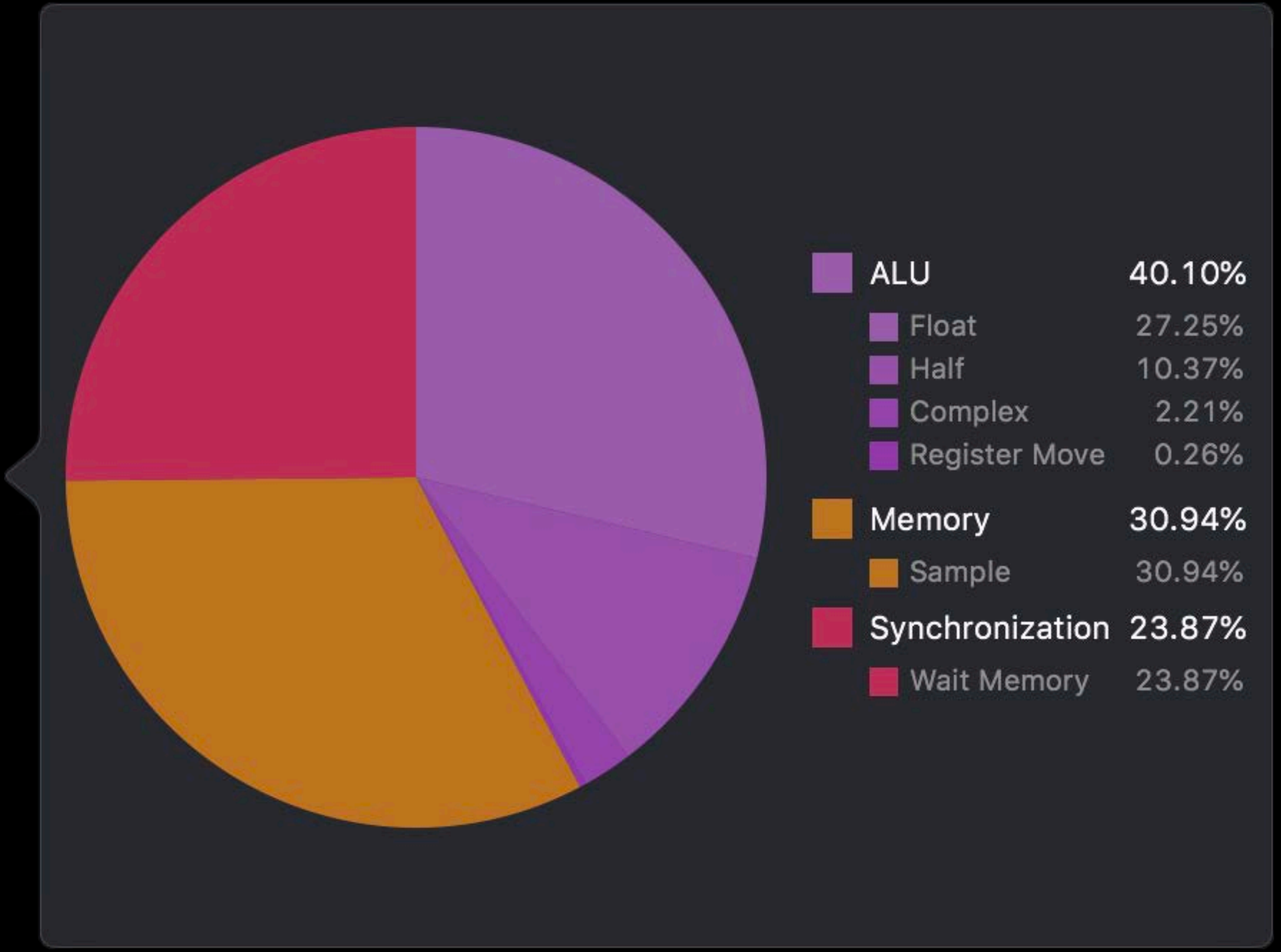
Get into shader debugger

```
float3 colorSample = colorMap.sample(colorSampler,           45.71%
    in.texCoord.xy*0.15 - uniforms.time[0] *
    0.0075).xyz;

float3 colorSample2 = colorMap.sample(colorSampler,           1.53%
    in.texCoord.xy*0.05).xyz;

float relativeHeight =                                        0.43%
    clamp(in.positionModelSpace.y + 0.5, 0.0, 1.0);
float2 relativeXZ = clamp((in.positionModelSpace.xz          14.65%
    + 1.5) / 3.0, 0.0, 1.0); // Thi should be 1.0 /
    2.0

float spatialVariation = sin(3 *                              4.27%
    (in.positionModelSpace.x +
    in.positionModelSpace.z));
```

Enhanced Shader Profiler for A11

NEW

Instruction category cost breakdown per line

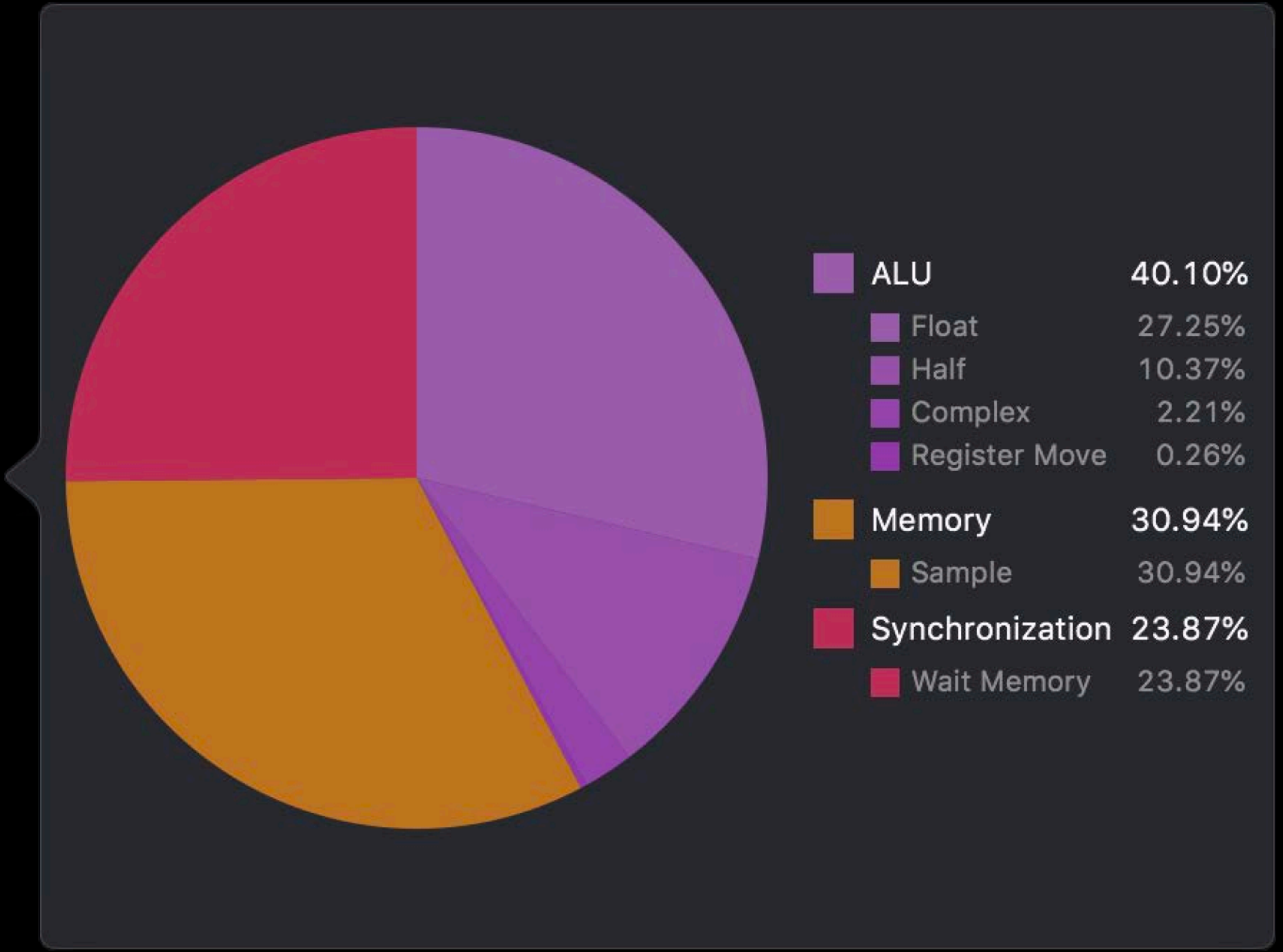


Enhanced Shader Profiler for A11

NEW

Instruction category cost breakdown per line

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

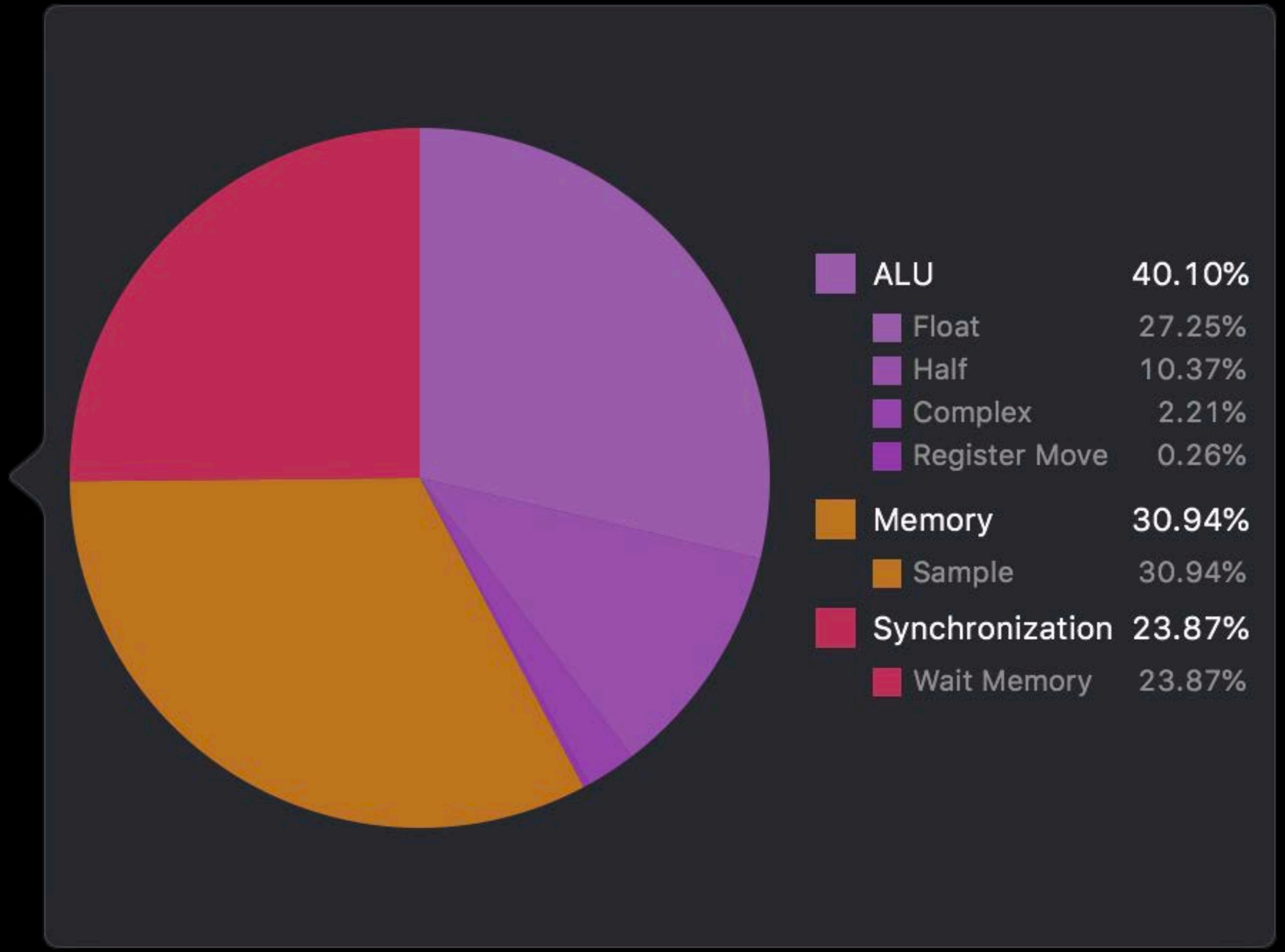


Enhanced Shader Profiler for A11

NEW

Instruction category cost breakdown per line

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



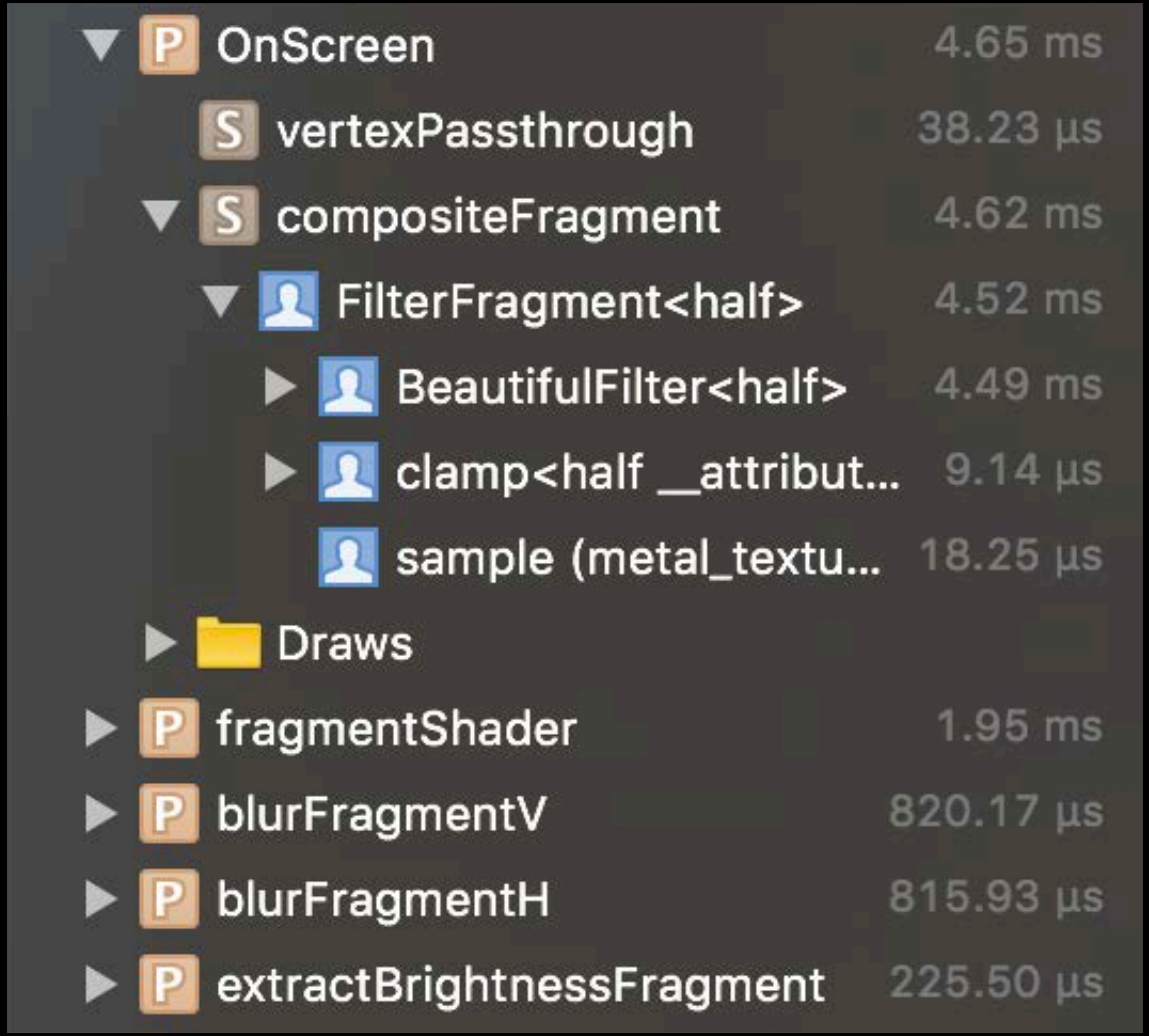
Enhanced Shader Profiler for A11

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



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	MetalDeferredLightingOSX
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