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

<table>
<thead>
<tr>
<th>Features</th>
<th>New Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step through API calls</td>
<td>Dependency viewer</td>
</tr>
<tr>
<td>Resource inspection</td>
<td>Geometry viewer</td>
</tr>
<tr>
<td>Shader edit and reload</td>
<td>Shader debugger</td>
</tr>
<tr>
<td>GPU counters</td>
<td>Enhanced shader profiler</td>
</tr>
<tr>
<td>Pipeline statistics</td>
<td></td>
</tr>
<tr>
<td>Integrated into Xcode</td>
<td></td>
</tr>
</tbody>
</table>
Metal Frame Debugger

- 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

NEW
Investigating Vertex Issues with Geometry Viewer
Things to Watch in Vertex Stage

Vertex inputs
Indices
Output
Geometry Viewer

Visualize the post-vertex transform data in 3D

Access to all vertex data

Per-draw call view
Visibly Wrong Triangles

<table>
<thead>
<tr>
<th>Index</th>
<th>Vertex</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Texture X</th>
<th>Texture Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>15564</td>
<td>19614</td>
<td>87.38</td>
<td>482.3</td>
<td>16.96</td>
<td>0.682</td>
<td>0.884</td>
</tr>
<tr>
<td>15565</td>
<td>19615</td>
<td>85.97</td>
<td>582.9</td>
<td>6.374</td>
<td>0.755</td>
<td>0.977</td>
</tr>
<tr>
<td>15566</td>
<td>19693</td>
<td>187.8</td>
<td>568.4</td>
<td>15.34</td>
<td>0.756</td>
<td>0.937</td>
</tr>
</tbody>
</table>

Color 1: Albedo + Shadow GBuffer
Visibly Wrong Triangles
Visibly Wrong Triangles
Visibly Wrong Triangles

<table>
<thead>
<tr>
<th>Index</th>
<th>in – uint Vertex</th>
<th>in – float3 position</th>
<th>in – float2 tex_coord</th>
<th>in – float2 normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>15564</td>
<td>19614</td>
<td>87.38 482.3 16.96</td>
<td>0.682 0.884 0.420</td>
<td>-0.305</td>
</tr>
<tr>
<td>15565</td>
<td>19615</td>
<td>85.97 502.9 6.374</td>
<td>0.755 0.877 0.361</td>
<td>-0.606</td>
</tr>
<tr>
<td>15566</td>
<td>19603</td>
<td>107.8 505.4 15.34</td>
<td>0.755 0.937 0.280</td>
<td>-0.647</td>
</tr>
</tbody>
</table>
### Visibly Wrong Triangles

<table>
<thead>
<tr>
<th>Index</th>
<th>in – uint Vertex</th>
<th>in – float3 position</th>
<th>in – float2 tex_coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>15564</td>
<td>19614</td>
<td>87.38 482.3 16.96</td>
<td>0.682 0.884 0.420 -0.302</td>
</tr>
<tr>
<td>15565</td>
<td>19615</td>
<td>85.97 502.9 6.374</td>
<td>0.755 0.877 0.361 -0.604</td>
</tr>
<tr>
<td>15566</td>
<td>19603</td>
<td>107.8 505.4 15.34</td>
<td>0.755 0.937 0.280 -0.647</td>
</tr>
</tbody>
</table>
Out of Frustum Objects
Out of Frustum Objects
Missing Triangles
Missing Triangles
Missing Triangles
Missing Triangles
Missing Triangles
Missing Triangles
Missing Triangles
### Missing Triangles

![Image of a 3D model with missing triangles]

### Table

<table>
<thead>
<tr>
<th>In - float3</th>
<th>Out - float4</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>position</td>
</tr>
<tr>
<td>7.3907</td>
<td>486.19</td>
</tr>
<tr>
<td>7.3907</td>
<td>486.19</td>
</tr>
<tr>
<td>4.7781</td>
<td>486.23</td>
</tr>
<tr>
<td>9.1662</td>
<td>475.32</td>
</tr>
</tbody>
</table>
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
Shader Debugger

New tool for debugging Metal shaders

Rich variable visualization across thousands of threads

Real data from GPU
Shader Debugger

New tool for debugging Metal shaders

Rich variable visualization across thousands of threads

Real data from GPU

Flexible stepping
Shader Debugger

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
<table>
<thead>
<tr>
<th>Vertex</th>
<th>out – float4 position</th>
<th>out – half3 color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-16.20 40.07 98.66 98.76</td>
<td>0.850 2.322 9.789</td>
</tr>
<tr>
<td>1</td>
<td>-15.87 39.84 98.66 98.76</td>
<td>0.850 2.322 9.789</td>
</tr>
</tbody>
</table>

**Perspective**: 0 0 0 1 0 0 0 0 0 0
**zNear**: 0.1
**zFar**: 100.0
**Field of View**: 65.0

**Debug**
<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread Position in Grid</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Threads per Grid</td>
<td>480</td>
<td>368</td>
<td>1</td>
</tr>
<tr>
<td>Thread Position in Threadgroup</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Threads per Threadgroup</td>
<td>16</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>Threadgroup Position in Grid</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Threadgroups per Grid</td>
<td>30</td>
<td>23</td>
<td>1</td>
</tr>
</tbody>
</table>
// Fragment shader for rendering fairies

#define PI 3.14159265358979323846

// Texture coordinates
vec2 texCoord = fragmentTextureCoord;

// Shader parameters
uniform vec4 colorMap; // Fairy color map
uniform float fairyOpacity; // Fairy opacity

// Main fragment shader
fragment fout fairy_frag {?
    vec4 color = vec4(0.5, 0.5, 0.5, 1.0);

    // Texture lookup
    vec4 colorSample = texture2D(colorMap, texCoord);
    color = vec4(colorSample.r, colorSample.g, colorSample.b, fairyOpacity);

    // Output color
    out vec4 fragColor = vec4(color.rgb, 1.0);
}

// End of fragment shader

declare (GL_FLOAT, GL_TRUE) precision mediump;

// Main entry function
main {?
    // Call the fragment shader
    float fragColor = 0.5;
    fragColor = float(fragColor);
    return float(fragColor);
}

// End of main entry function
Inspecting Variables

Just go the source line

```cpp
class MagFilter {
    public:
        static const half4 mag_filter::linear();

    half2 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).x;
    float3 normal = calcNorm(tangent_normal, in);

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

class Sampler {
    public:
        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;
```
Inspecting Variables

Just go the source line

• The side bar for the modified variables

```cpp
map_filter::linear;

half2 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_function::less);

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

FragOutput output;
```
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
See Variables in Context

```c
float4 grad = gradient(color1, color2, input);
```

```c
eye_normal = normalize(eye_normal);

const expr 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_comp;
```
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

- G-buffer Creation: 1.25 ms
- Sky: 661.39 µs
- Shadow Gen: 409.31 µs
- Deferred Directional Lighting: 362.65 µs
- Light: 293.16 µs
- Point Light Mask: 276.72 µs
- Fairy Drawing: 13.37 µs
Shader Profiler

Provides per-pipeline timing information

- G-buffer Creation: 1.25 ms
- gbuffer_vertex: 1.02 ms
- gbuffer_fragment: 1.21 ms
- Sky: 661.39 µs
- Shadow Gen: 409.31 µs
- Deferred Directional Lighting: 362.65 µs
- Light: 293.16 µs
- Point Light Mask: 276.72 µs
- Fairy Drawing: 13.37 µs
Shader Profiler

Provides per-pipeline timing information

- G-buffer Creation: 1.25 ms
- gbuffer_vertex: 1.02 ms
- gbuffer_fragment: 1.21 ms

- Draws:
  - 42 drawIndexedPrimitives: 1.01 ms
  - 46 drawIndexedPrimitives: 225.51 µs
  - 50 drawIndexedPrimitives: 7.81 µs

- Sky: 661.39 µs
- Shadow Gen: 409.31 µs
- Deferred Directional Lighting: 362.65 µs
- Light: 293.16 µs
- Point Light Mask: 276.72 µs
- Fairy Drawing: 13.37 µs
Shader Profiler

Provides per-pipeline timing information

Per-line execution cost (iOS and tvOS)

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

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

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

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

Provides per-pipeline timing information

Per-line execution cost (iOS and tvOS)

Shader edit and reload

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

45.71%

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

1.53%

float relativeHeight = clamp(in.positionModelSpace.y + 0.5, 0.0, 1.0);

0.43%

float2 relativeXZ = clamp((in.positionModelSpace.xz + 1.5) / 3.0, 0.0, 1.0); // This should be 1.0 / 2.0

14.66%

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

4.27%
Enhanced Shader Profiler for A11

Instruction category cost breakdown per line

- ALU: 40.10%
- Memory: 30.94%
- Synchronization: 23.87%
- Float: 27.25%
- Half: 10.37%
- Complex: 2.21%
- Register Move: 0.28%
Enhanced Shader Profiler for A11

Instruction category cost breakdown per line

- **ALU** — Float, half, and complex
- **Memory** — Sample, load, and store operations
Enhanced Shader Profiler for A11

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

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
Access to Shader Sources

New Metal Compiler option

Xcode project
• "Yes, include source code" from build settings

Command line
• "-MO" compiler option
Recap

Geometry viewer
Recap

Geometry viewer

Shader debugger
Recap

Geometry viewer

Shader debugger

Enhanced shader profiler
More Information

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

<table>
<thead>
<tr>
<th>Event</th>
<th>Location</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metal Game Performance Optimization</td>
<td>Hall 1</td>
<td>Fri 10:00 AM</td>
</tr>
<tr>
<td>Metal Debugging and Profiling Lab</td>
<td>Technology Lab 5</td>
<td>Fri 12:00 PM</td>
</tr>
</tbody>
</table>