UE4
Mobile Rendering Overview

Content

- RHI
- Shader crosscompiler
- Forward renderer
- HDR rendering
- Performance
- Vendor specific tools

UE4

- UE4 is a cross platform engine
- Runs on many platforms
- Desktops, Consoles, Mobile devices, Web
- Extremely important that we can develop features once and have them work on all platforms

Game Sim

World

PrimitiveComponent
LightComponent
Etc

Tick
SimulatePhysics
Etc

Renderer

Scene

PrimitiveInfo
LightInfo
Etc

RenderBasePass
PostProcess
Etc

CROSS PLATFORM

RHI

Shaders
Textures
VertexBuffers
Etc

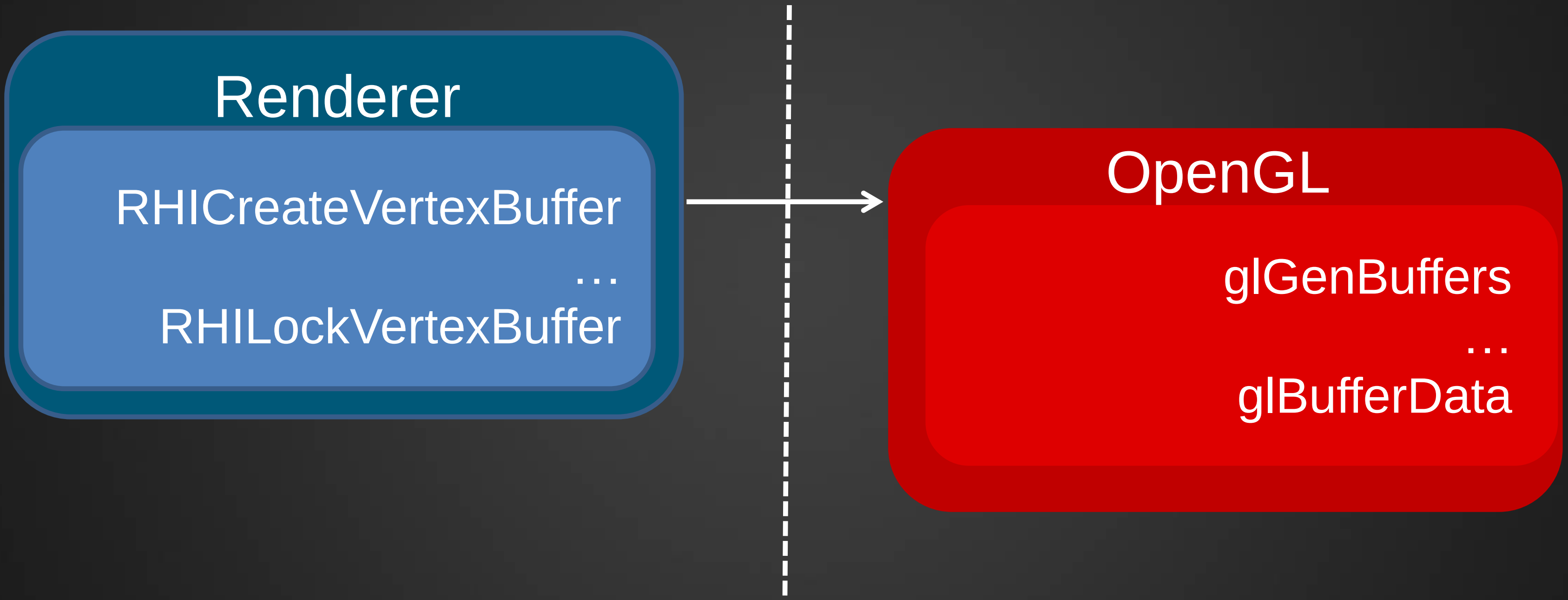
PLATFORM DEPENDENT

Rendering Hardware Interface

RHI implementations:

- D3D11
- OpenGL
- Metal
- Gnm (PS4)
- Experimental:
  - D3D12
  - Vulkan

Rendering Hardware Interface

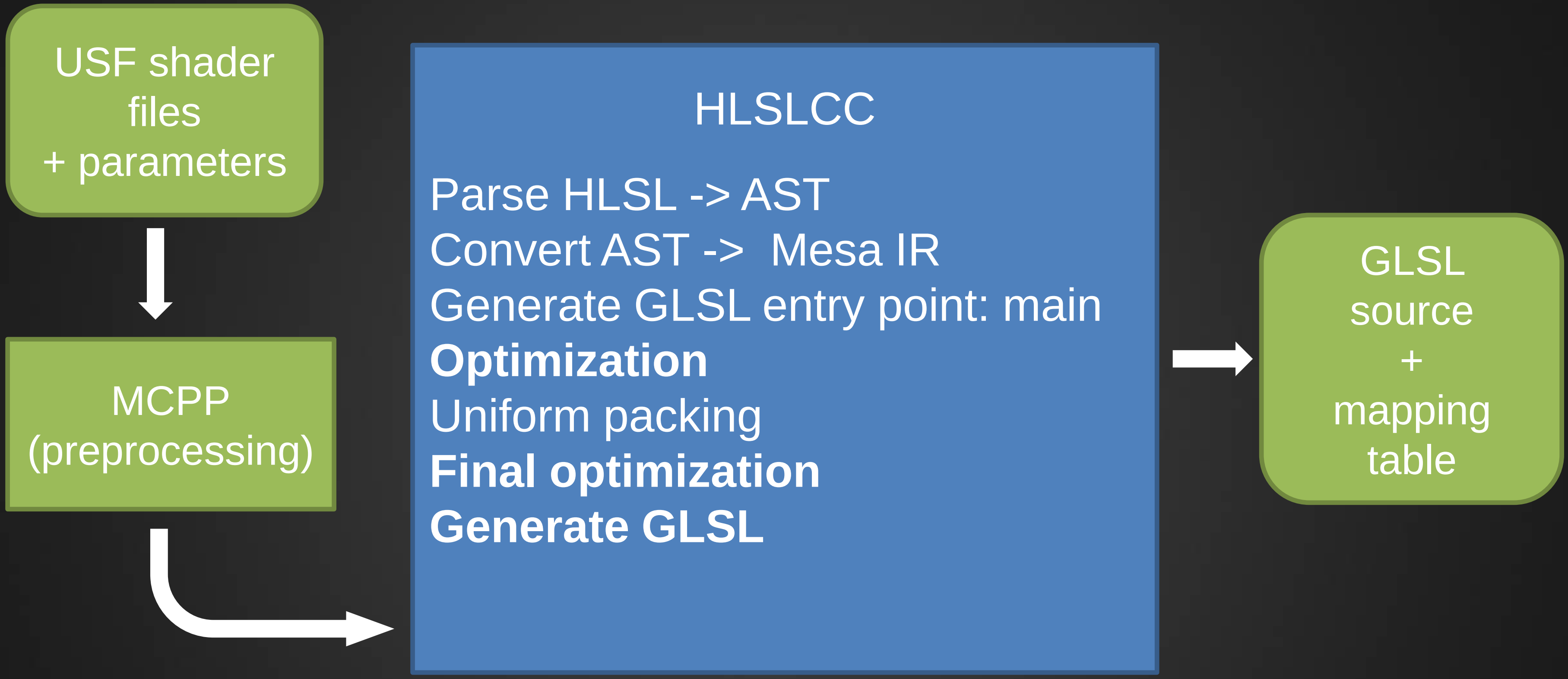


OpenGL RHI

- Supports multiple GL versions
  - GL4, GL3, ES2, ES3
- Most features have corresponding SupportXXX() function
  - bool FOpenGL::SupportsHarwareInstancing()
- Shares as much functionality as possible
- Much of the API is same between versions
  - RHICreateVertexBuffer -> glGenBuffers + glBufferData
  - RHIDrawIndexedPrimitive -> glDrawElements

Shaders

- UE4 has large existing HLSL-like shader code base
- We do not want to write shaders for each platform
- Want to compile and validate our shaders offline
- Need to create metadata used by renderer at runtime
  - Which textures bound to which indices
  - Which uniforms are used and need to be uploaded to GPU


USF shader files
+ parameters
MCPP
(preprocessing)
HLSLCC
Parse HLSL -> AST
Convert AST -> Mesa IR
Generate GLSL entry point: main
Optimization
Uniform packing
Final optimization
Generate GLSL
GLSL source
+
mapping table

GLSL backend

- Generates GLSL code for a requested platform
  - GL4, ES2
- Target validation
- Platform specific workarounds
- Places GL extensions depending on features used in the shader

```
if (bUsesDepthbufferFetchES2)
{
    ralloc_asprintf_append(buffer,
      "#extension GL_ARM_shader_framebuffer_fetch_depth_stencil : enable\n");
}
```

Benefits of cross compilation

- Write shaders once, they mostly work everywhere
- Offline validation and shader metadata
- Platform specific workarounds in the compiler itself
- Enable r.DumpShaderDebugInfo to see output of shader compilation
  - Will dump processed HLSL and cross compiled GLSL into project temp folder

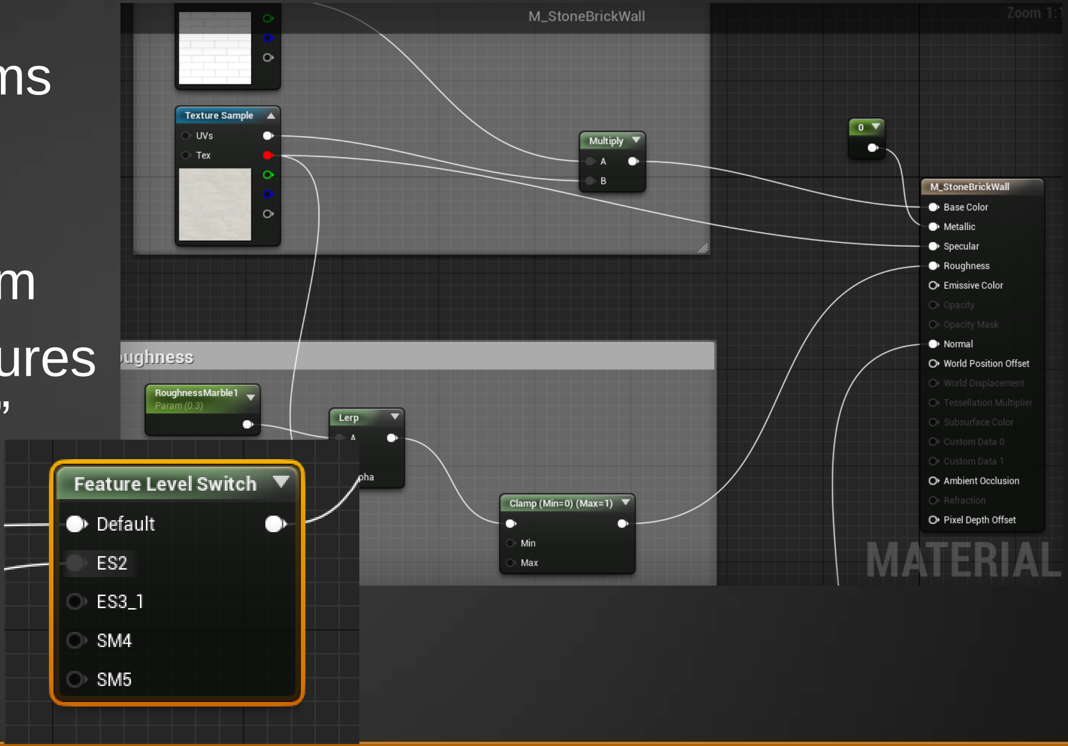
- Allows mobile preview in Editor

Authoring consistency

- Want same controls on high-end and mobile
- Authoring environment for both platforms
  - Physically based shading model
  - High dynamic range linear color space
  - High quality post processing
- Preview content in editor with mobile feature set
  - Uses forward render, same as on mobile device
  - Shaders re-compiled with mobile specific defines

Consistency: Material Editor

- One material for all platforms
- Material editor will warn if some features are not supported on target platform
- Gate optional material features with “Feature Level Switch”



Consistency: Physically Based Shading

- Same material model as on high-end
  - Intuitive material controls (BaseColor, Metallic, Roughness, Specular)
  - See: “Real Shading in Unreal Engine 4” (Brian Karis)

- Mobile adjustments
  - Analytic approximation of Environment BRDF (ALU instead of TEX)
  - Normalized Phong specular distribution (faster and still energy conserving)
  - See: https://www.unrealengine.com/blog/physically-based-shading-on-mobile

Consistency: Physically Based Shading

ENVBRDF used on high-end

ENVBRDF approximation on Mobile

Consistency: Rendering

- Not always possible to implement rendering features the same way on all platforms
- Recently added Deferred decals, Refraction, GPU particles to Mobile, etc
- Most of these rendering features have limitations on mobile platforms
  - Decals are unlit and support only small subset of blending options
  - GPU particles do not support collisions

Sharing rendering code

- Always share as much code as possible
- Forward renderer has very few code that is forward specific
- Most of rendering features use same code on forward and deferred path
- This allows to reduce support cost
- When someone adds an improvement to a rendering feature it affects all platforms

Scene rendering

```
FSceneRenderer* CreateSceneRenderer(const FSceneViewFamily* InViewFamily)
{
	if (FeatureLevel <= ERHIFeatureLevel::ES31)
	{
		return new FForwardShadingSceneRenderer(InViewFamily);
	}
	else
	{
		return new FDeferredShadingSceneRenderer(InViewFamily);
	}
}
```

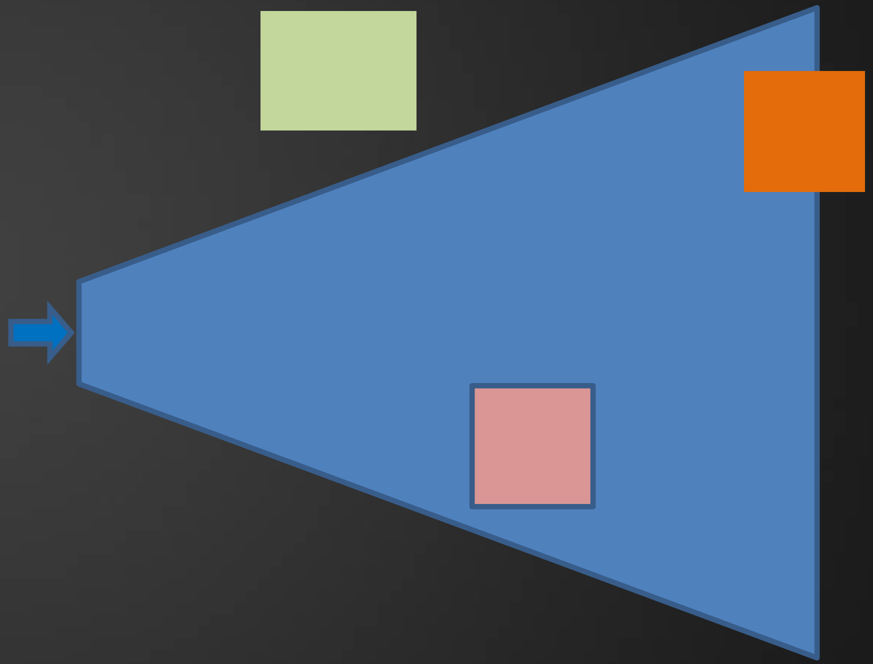
Editor can switch FeatureLevel on the fly

Forward renderer (mobile)

1. Views setup
2. GPU particles simulation
3. Shadow maps
4. Base pass
5. Deferred decals
6. Modulated shadows projections
7. Translucency
8. Post-process
9. HUD

1. View setup

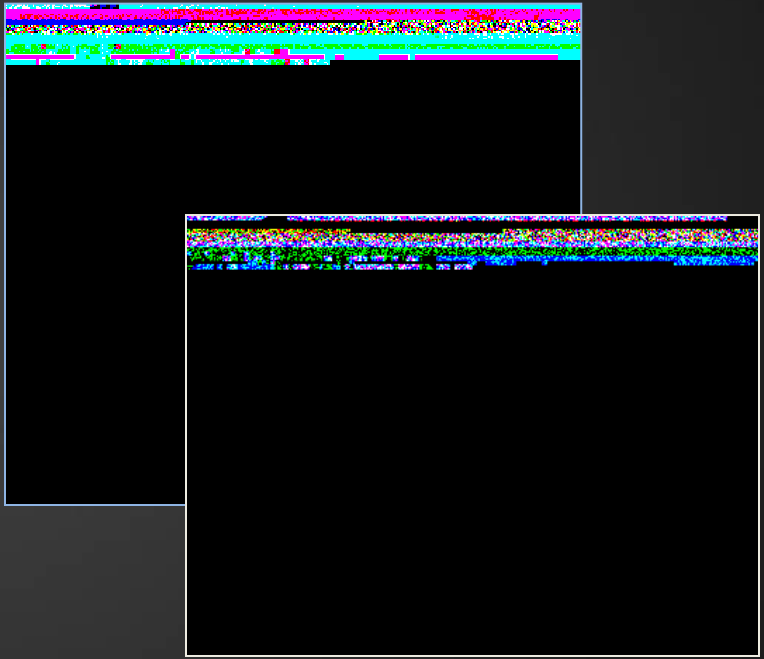
- Find out all visible objects
  - Frustum culling
  - Distance culling
  - Precomputed visibility
- Find out all visible shadows
- Gather dynamic mesh elements
- Update view uniform buffer



2. GPU particles simulation

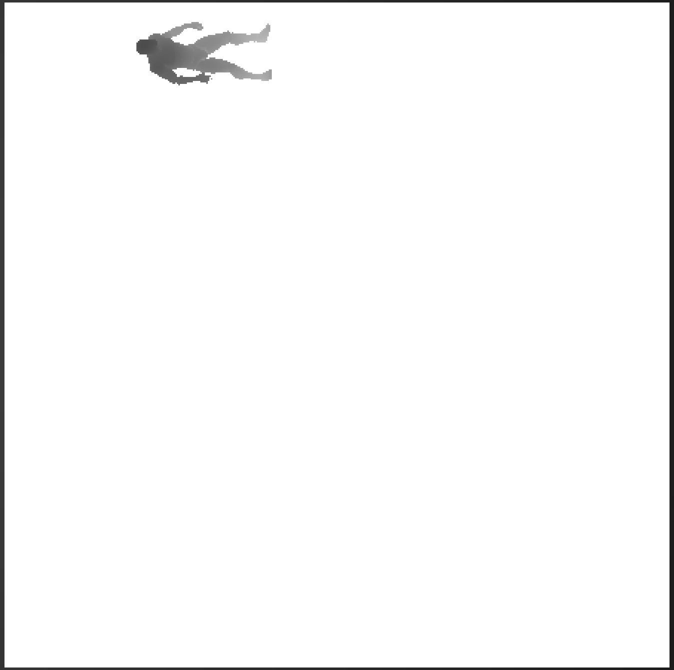
Requires ES31\Metal features

- Simulates particle physics on GPU
- Writes particle Position to 128bpp target
- Writes particle Velocity to 64bpp target



3. Shadow maps

- Setup depth render target
- Find out which objects need shadows
- Render them using main light view
- Shadow map used later
  - base pass
  - modulated shadow projections



Base pass (setup)

- Assign primitives to a specific list depending on shading properties
  - Unlit
  - Distance field shadows + LM
  - Distance field shadows + LM + CSM
  - ETC
- Primitives in each list grouped by material, vertex factory to reduce OpenGL state changes

4. Base pass drawing

- Drawing order depends on device
  1. Draw without any reordering
  2. Reorder meshes front to back in each list
  3. Reorder meshes front to back across all lists
- ImgTec has “Hidden Surface Removal”
  - no distance sorting required (1)
- Others - third (3) option by default
- Can be tweaked depending on your content
  - r.ForwardBasePassSort = x

5. Deferred decals

- Requires scene depth fetch
  - Implementation depends on supported extensions
  - GL_ARM_shader_framebuffer_fetch_depth_stencil
  - GL_EXT_shader_framebuffer_fetch
  - GL_OES_depth_texture
    - depth buffer resolve
- Supports “Receives Decals” flag
  - Stencil operations
- Does not support lighting

6. Modulated shadow projections

- Similar to deferred decals
- Does not blend well with static lighting
- CSM is a better option (in 4.12)

7. Translucency

- Draw primitives with refraction
  - Requires full copy of scene color
- Draw primitives with translucent materials
  - Usually particles

8. Postprocess

- Only when HDR is enabled
- Requires several passes depending on what effects are used
- Tonemapper pass at the end
  - Maps HDR color to {0, 1} range and writes it to backbuffer

9. HUD

- Draw UI elements to backbuffer
  - Slate
  - Canvas
- Swap backbuffer

OpenGL state caching

- OpenGL RHI caches state
- Most of calls to RHI just setup pending state inside RHI
- Commits pending state on DrawPrimitive calls or Clears
- Need to be careful about changing GL state outside of RHI
  - For example drawing something from Java
  - Make sure to save and then restore all changed state
  - Recently fixed OpenGL state bug within our java MoviePlayer

Rendering

LDR

- Render directly to backbuffer
- In gamma space
  - Blending might look a bit wrong
- No postprocessing
- Fastest path
- Used by default for GearVR apps

HDR

- Render to a texture (SceneColor)
- Linear space
  - Correct blending
- Postprocessing
  - Bloom, DOF, Tonemapping
- Enables additional features
  - Deferred decals, Refraction

HDR rendering – FP16

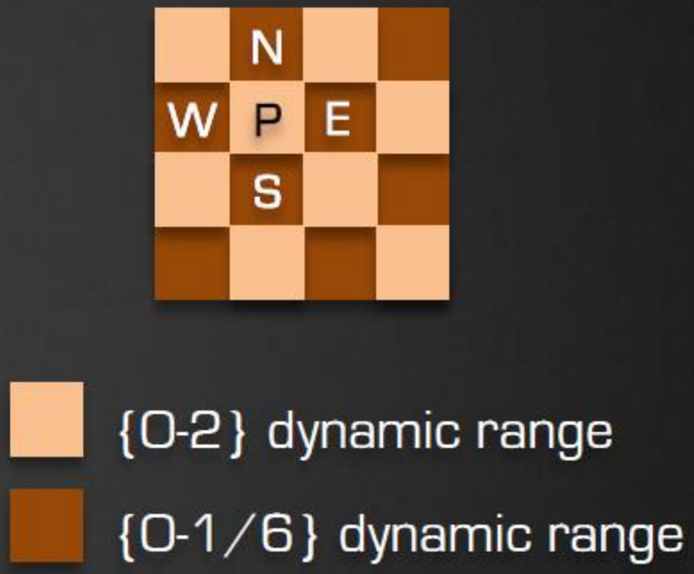
- Requires half-float color buffer support
  - GL_EXT_color_buffer_half_float
- Color in RGB channels
- Depth in Alpha channel
- In case device supports frame buffer fetch - uses more optimal path with less render target switches
  - All supported iOS devices
  - Adreno 3xx and up (Nexus5)
  - Mali 8xx (Galaxy S6, S7)

HDR rendering - 32bpp Encoding

- Uses 8bit per channel color buffer
- Scale RGB values to {0-1} range and store exponent in Alpha channel
- Supports {0, 1024} dynamic range
- Requires frame buffer fetch extension
  - GL_EXT_shader_framebuffer_fetch
  - GL_ARM_shader_framebuffer_fetch
- Custom blending with frame buffer fetch
- Usually faster than FP16 path
  - Mali 7xx (Galaxy Note4)
  - Possible to support in iOS devices

HDR rendering - Mosaic Encoding

- When device has no FP16 color buffer and no frame buffer fetch
- Resolution limited to 1024
- Limited post-processing (Bloom, Tonemapping)
- Supports {0, 2} dynamic range
- Simulates 12-bit linear color buffer
- Demosaic in tonemapper



Performance scaling

LDR (Low Dynamic Range)

- Fastest mode
- Use when you don’t need lighting or post-process effects
- Disable “Mobile HDR” in Rendering section in your Project Settings

Performance scaling

Basic Lighting

- Allows HDR lighting and some post-process effects
- Use only static lights
- Use only fully rough materials, not shiny (specular)
- Disable Bloom and anti-aliasing

Performance scaling

Full HDR Lighting

- High-quality lighting with best support for normal maps
- Realistic specular reflections on surfaces with per-pixel roughness
- Use only static lights
- Bloom and anti-aliasing are recommended
- Place reflection captures carefully for best results

Performance scaling

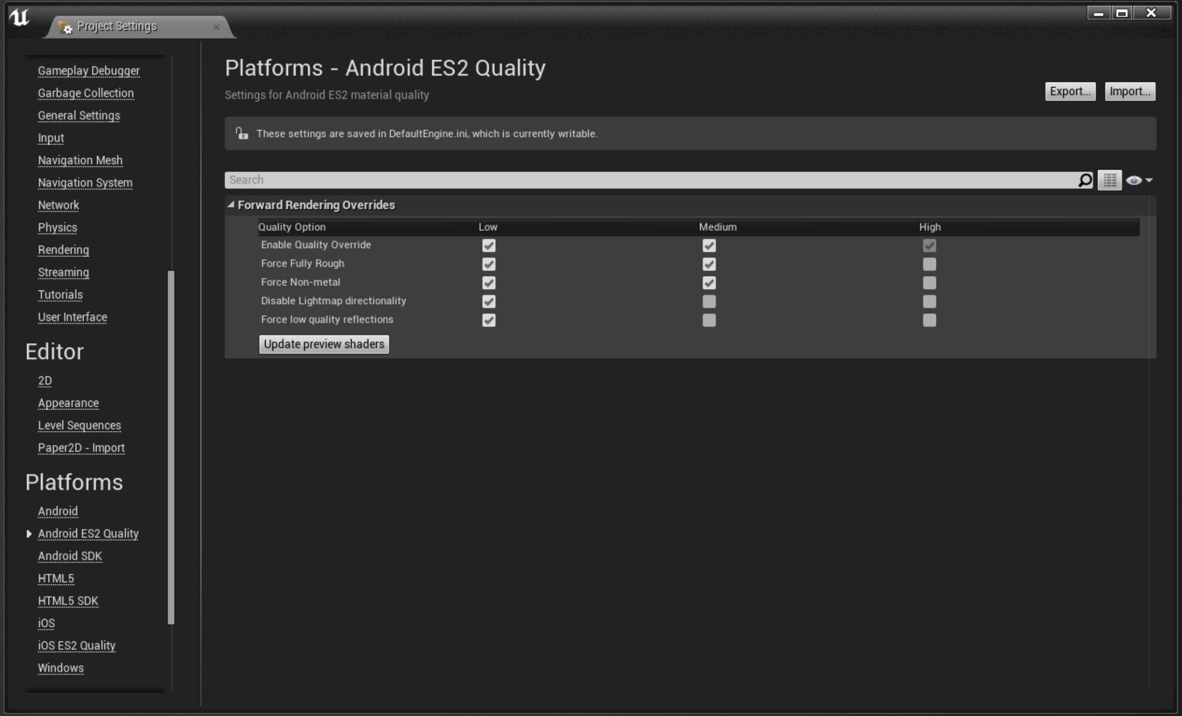
Full HDR Lighting with per-pixel lighting from the Sun

- Specify one directional light as stationary (the Sun)
- All other lights are static
- High-quality distance field shadows

Performance scaling

- Shading quality
- 3 sets of shaders
- Select at app startup depending on device

- Preview in Editor: Settings->Material Quality Level



Performance

- Use as few unique materials as possible (use Material instances)
  - Minimizes state switches
- Use as few draw calls as possible
  - Merge static geometry in the Editor (use Merge Actors tool)
  - Use precomputed visibility
- Dynamic batching - does not exists in engine
  - CCP added it for Gunjack
- Can use Unlit materials and fake lighting

Debugging mobile issues

- Make sure problem does not exists on PC (Mobile preview)
  - It's always easier to debug on PC than on mobile device
- Enable ENABLE_VERIFY_GL (disabled by default)
- Enable framebuffer checks (enabled only in DEBUG configuration)
- Shaders: FORCE_FLOATS if you suspect precision issues
  - Will force highp floats in pixel shaders
- Watch log output, as it may report shader compilation errors
- Use vendor specific tools

Watch out: precision

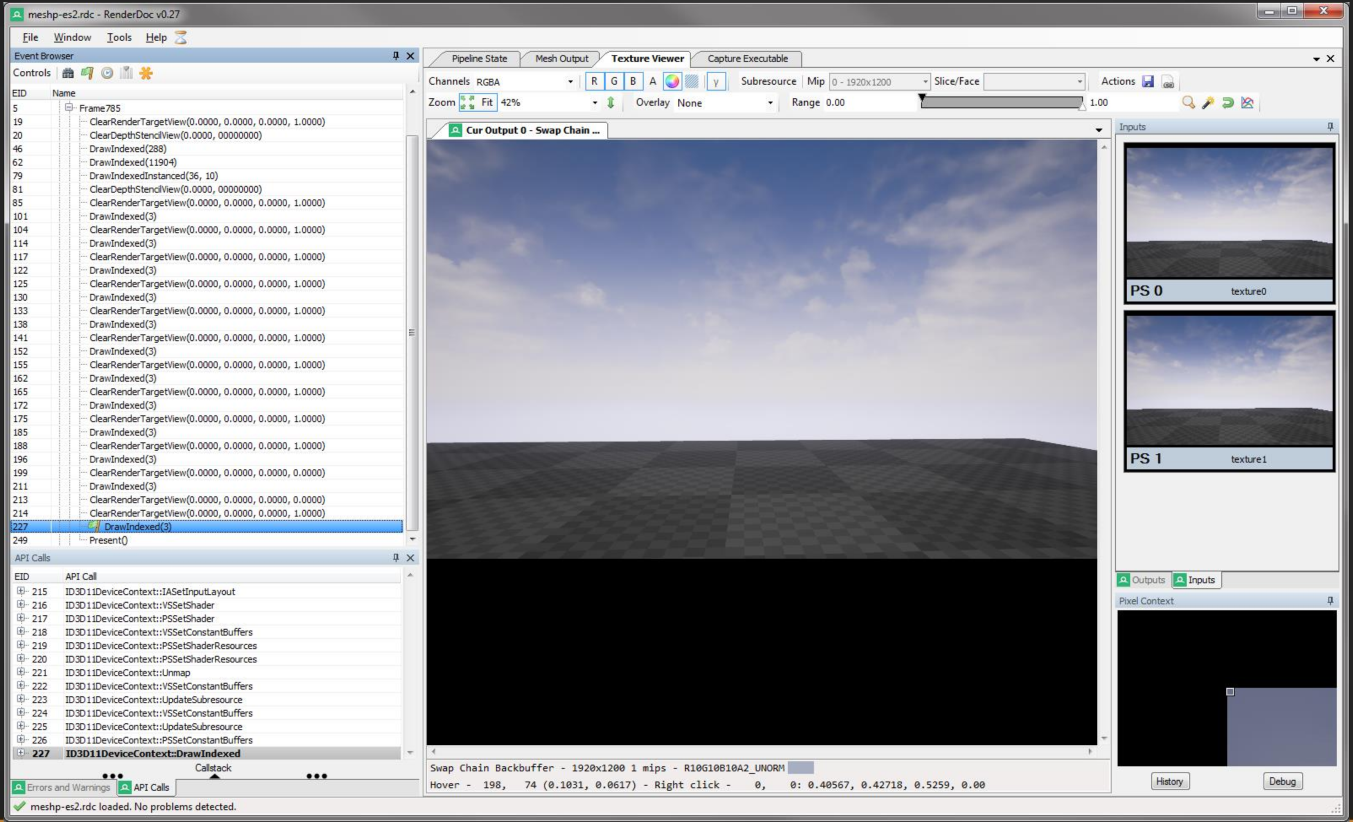
- Be careful when using mediump with functions like
  - length()
  - normalize()
  - distance()
- mediump is usually in: -2^14 … 2^14 range
  - length = sqrt(v.x^2+ v.y^2+ v.z^2)
- Make sure that vector magnitude is less 2^7 (128)
- length (vec3(0, 129, 0)) may produce INF on some devices

Platform-Specific Profiling

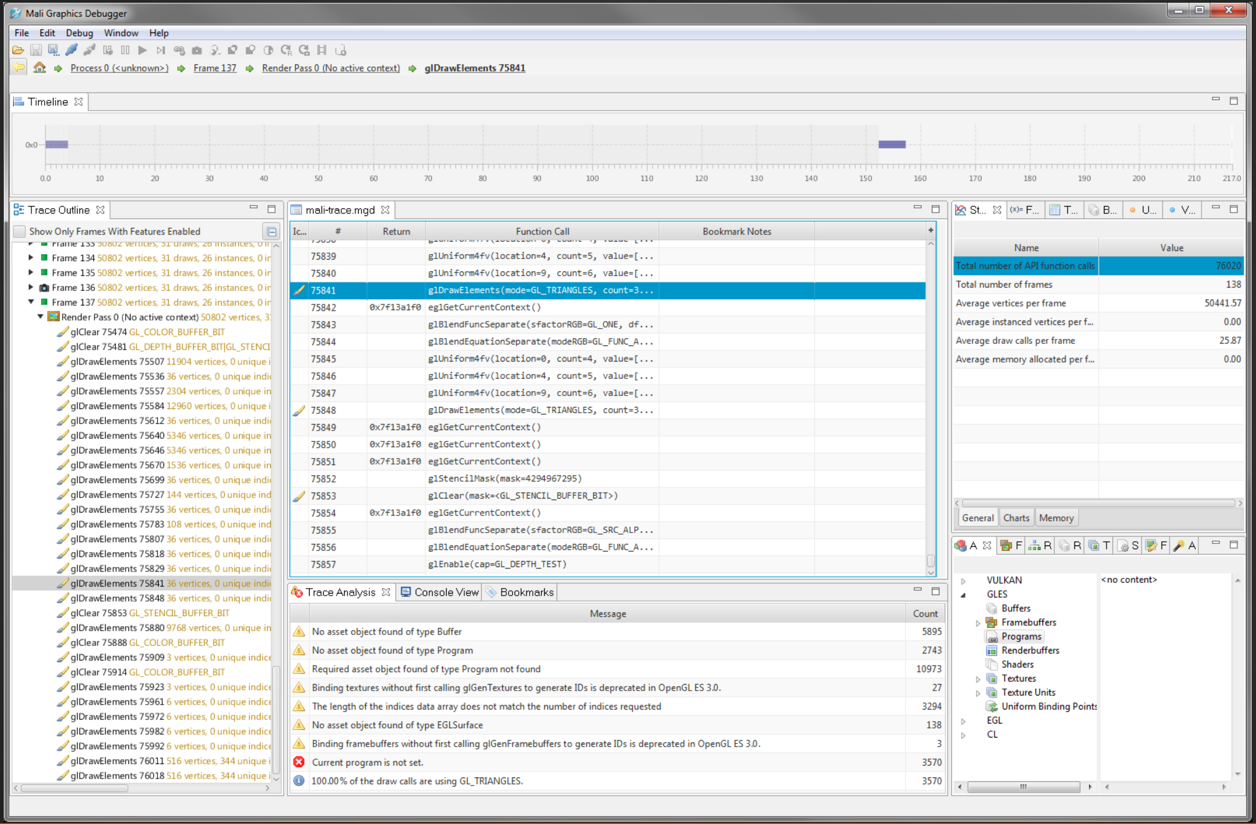
- Each GPU family has their own profiling tools
  - Apple: Xcode GL Debugger (and Metal)
  - Qualcomm: Adreno Profiler
  - NVIDIA: Tegra Graphics Debugger
  - ImgTec: PVRTune, PVRTrace
  - ARM: Mali Graphics Debugger

- Intel GPA works on several platforms

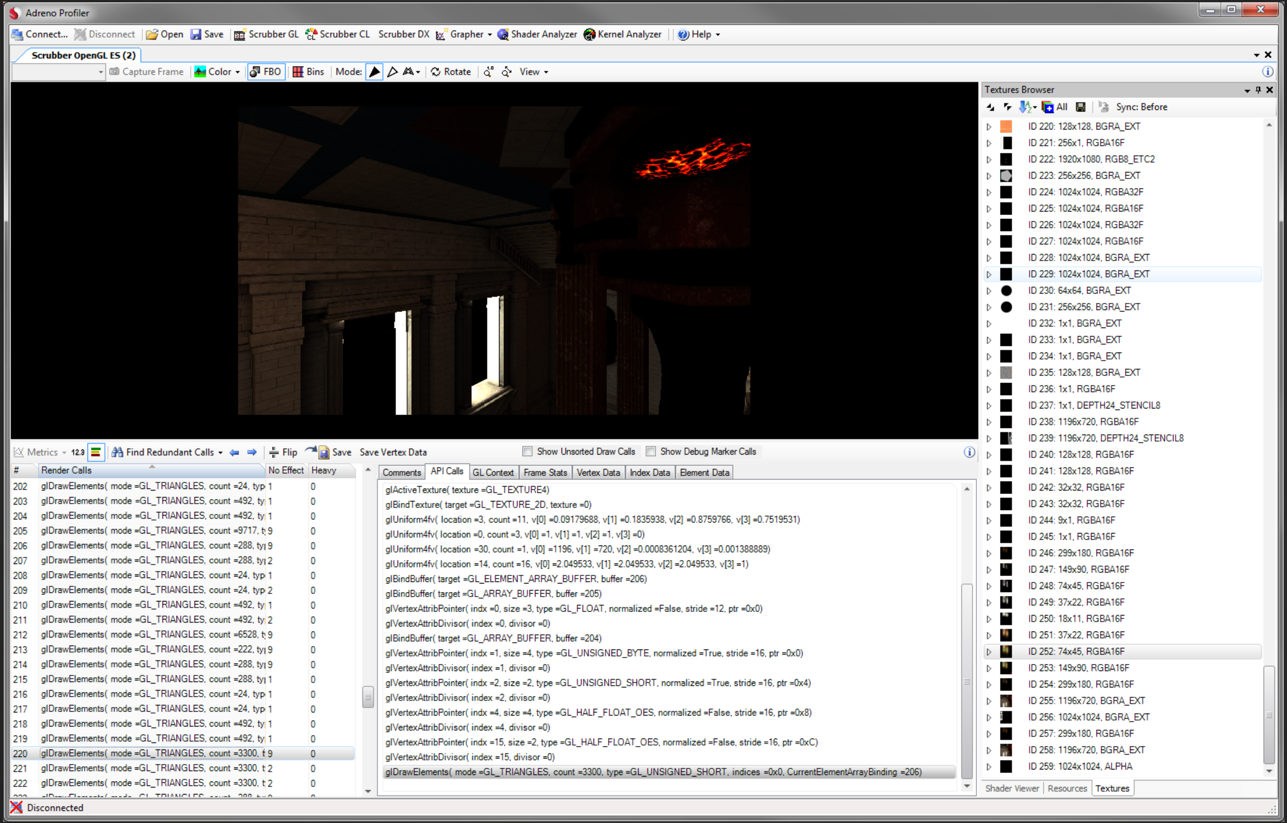
RenderDoc


meshp-es2.rdc - RenderDoc v0.27
File Window Tools Help
Event Browser
Controls
EID Name
5 Frame785
19 ClearRenderTargetView(0.0000, 0.0000, 0.0000, 1.0000)
20 ClearDepthStencilView(0.0000, 00000000)
46 DrawIndexed(288)
62 DrawIndexed(11904)
79 DrawIndexedInstanced(36, 10)
81 ClearDepthStencilView(0.0000, 00000000)
85 ClearRenderTargetView(0.0000, 0.0000, 0.0000, 1.0000)
101 DrawIndexed(3)
227 DrawIndexed(3)
249 Present()
API Calls
EID API Call
215 ID3D11DeviceContext::IASetInputLayout
216 ID3D11DeviceContext::VSSetShader
217 ID3D11DeviceContext::PSSetShader
218 ID3D11DeviceContext::VSSetConstantBuffers
219 ID3D11DeviceContext::PSSetShaderResources
220 ID3D11DeviceContext::PSSetShaderResources
221 ID3D11DeviceContext::Unmap
222 ID3D11DeviceContext::VSSetConstantBuffers
223 ID3D11DeviceContext::UpdateSubresource
224 ID3D11DeviceContext::VSSetConstantBuffers
225 ID3D11DeviceContext::UpdateSubresource
226 ID3D11DeviceContext::PSSetConstantBuffers
227 ID3D11DeviceContext::DrawIndexed
Callstack
Errors and Warnings API Calls
meshp-es2.rdc loaded. No problems detected.
Pipeline State Mesh Output Texture Viewer Capture Executable
Channels RGBA R G B A Subresource Mip 0 - 1920x1200 Slice/Face Actions
Zoom Fit 42% Overlay None Range 0.00 1.00
Cur Output 0 - Swap Chain ...
Inputs
PS 0 texture0
PS 1 texture1
Outputs Inputs
Pixel Context
History Debug
Swap Chain Backbuffer - 1920x1200 1 mips - R10G10B10A2_UNORM
Hover - 198, 74 (0.1031, 0.0617) - Right click - 0, 0: 0.40567, 0.42718, 0.5259, 0.00

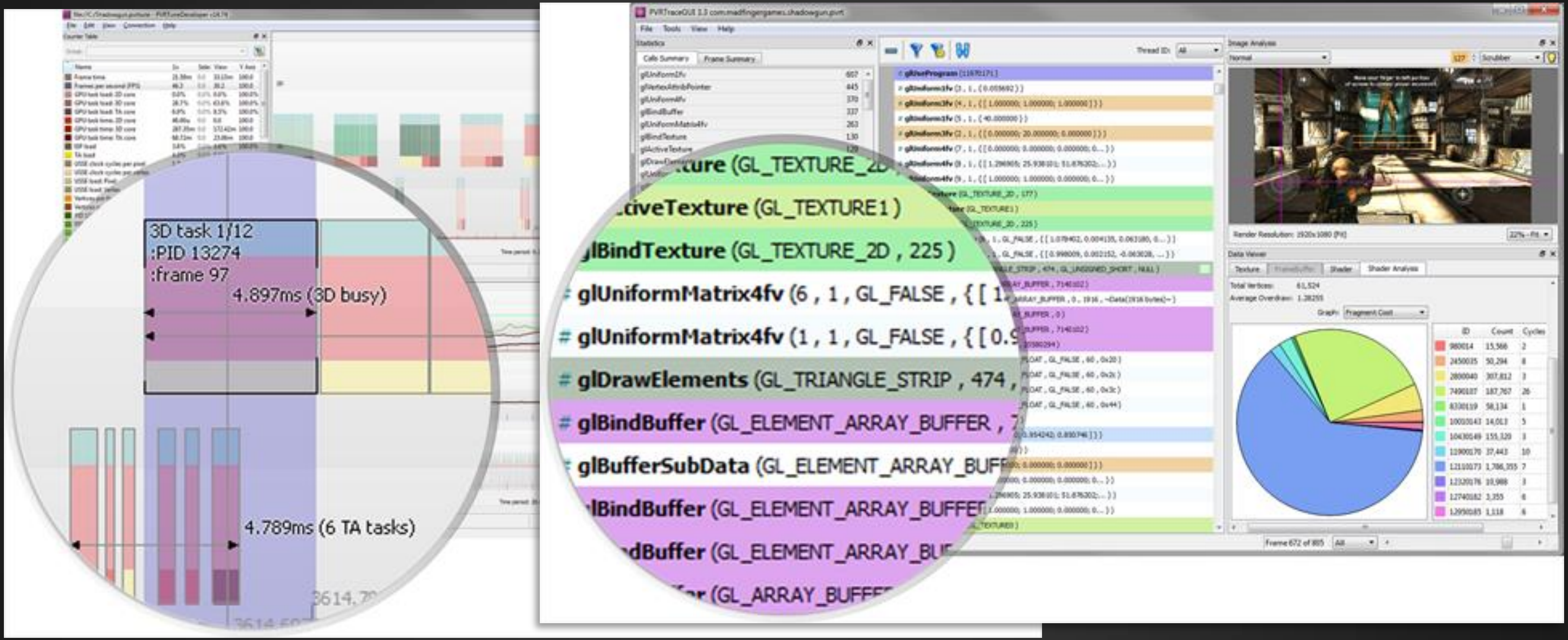
Mali Graphics Debugger



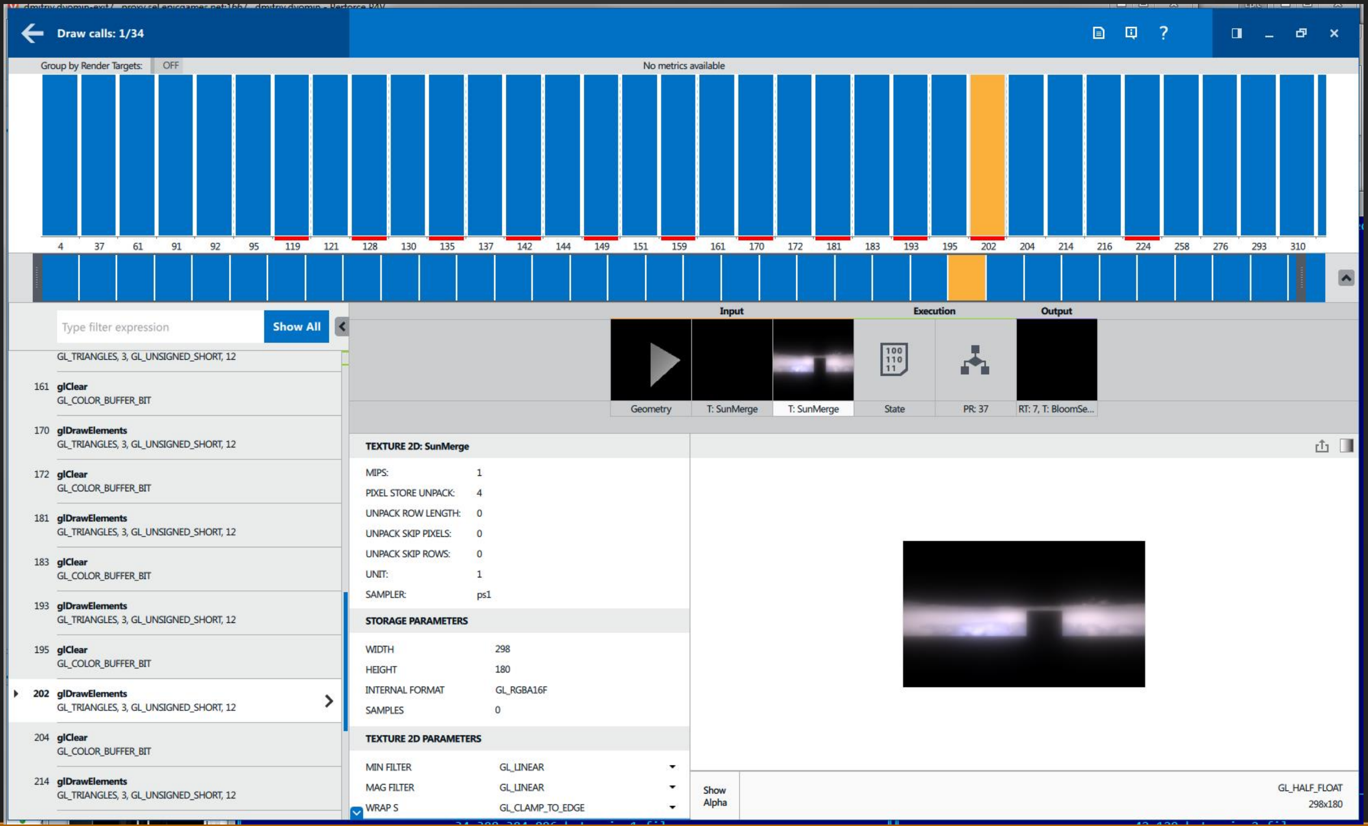
Adreno Profiler

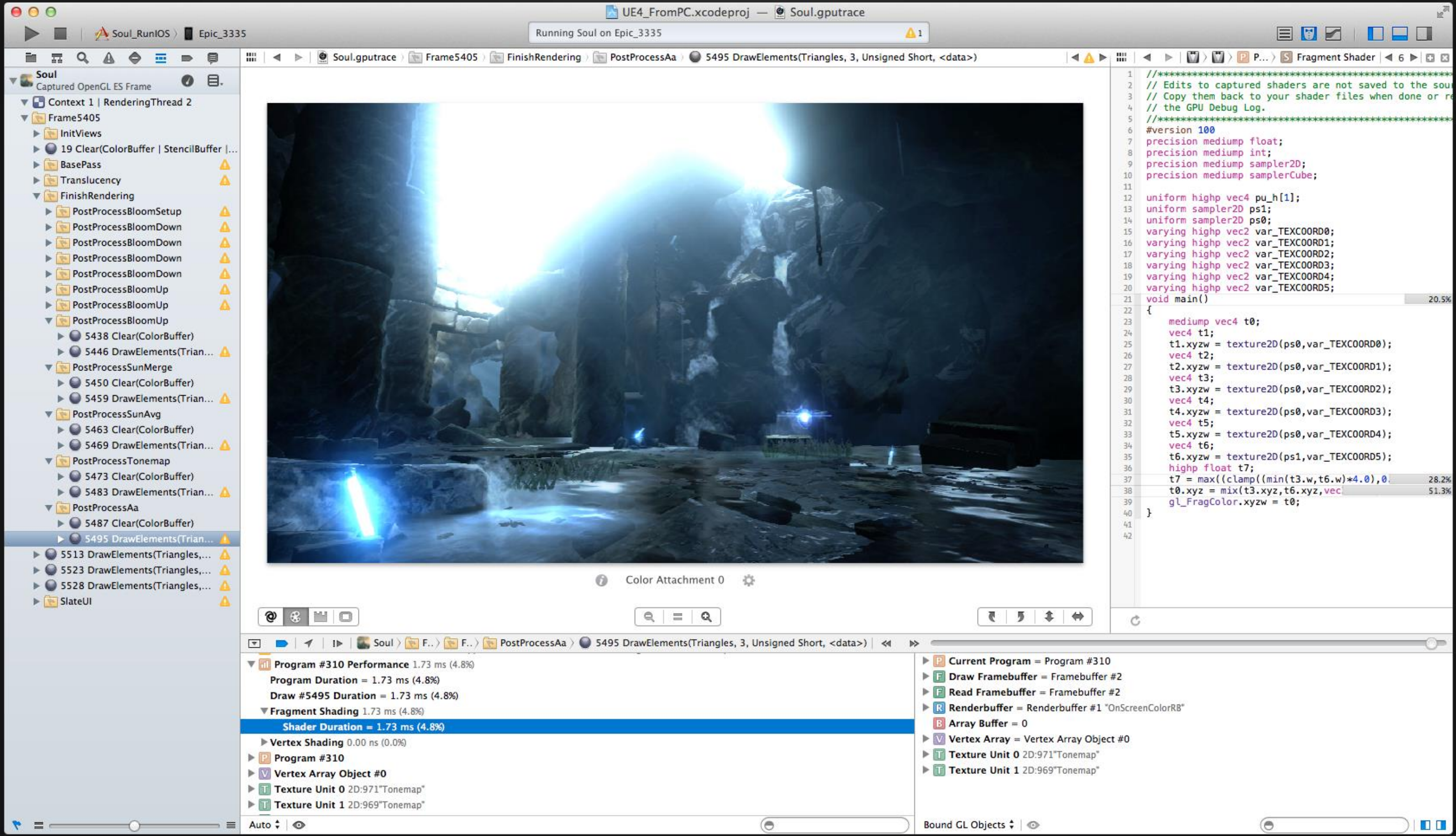


ImgTec PVRTune and PVRTrace



Intel Graphics Frame Analyzer for OpenGL





UNREAL ENGINE

UNREAL SUMMIT 2016

Done!

- Questions?