UE4 Mobile Performance

Niklas Smedberg

Senior Engine Programmer, Epic Games

Content

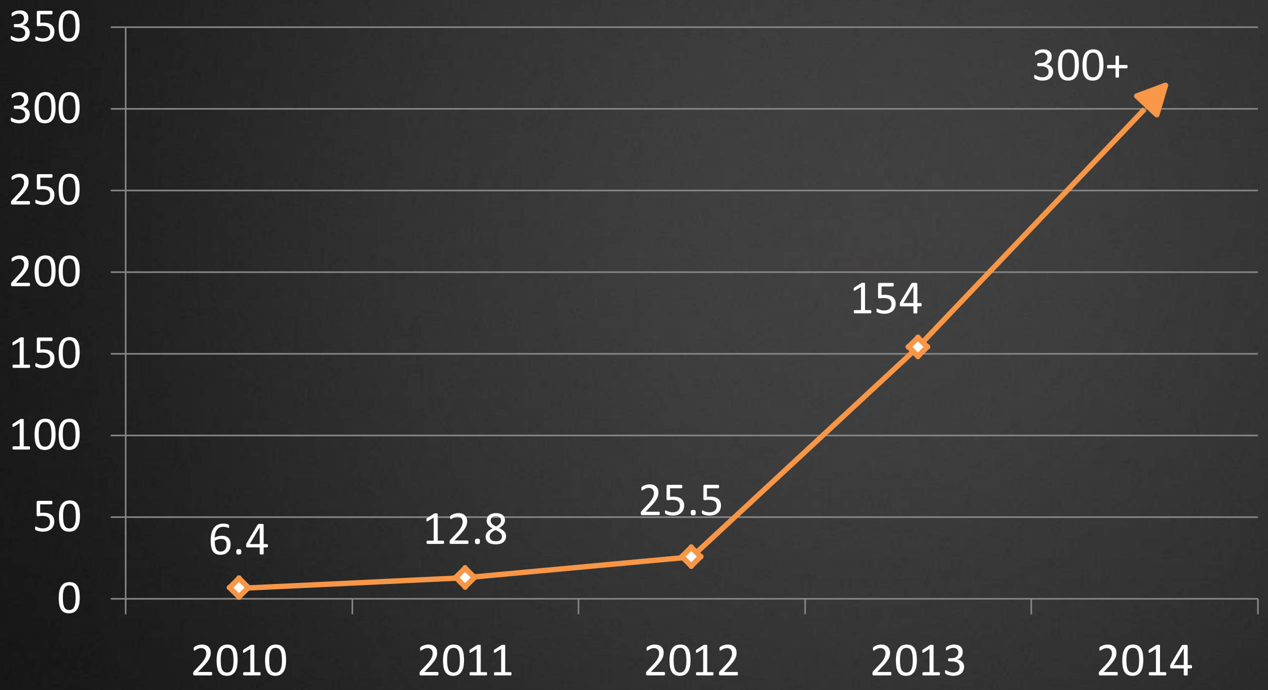
- Part 1: Understanding mobile performance
  - Mobile GPU Hardware
  - Thermal limits
  - Performance guidelines

- Part 2: Adapt and conquer
  - Cross-platform profiling
  - Platform-specific profiling
  - Scaling your game based on device

Part 1: Understanding Mobile Performance

- Mobile hardware is evolving at a crazy rapid rate
- Next-generation mobile GPUs:
  - Fully featured (DirectX 11)
  - Peak performance comparable to Xbox 360 and PS3
    - 300+ GFLOPS and 26 GB/s
  - Able to run full UE4 desktop high-end rendering pipeline (e.g. NVIDIA K1)
- Phone users upgrade hardware very frequently
  - But tablet users don’t
  - Also, new large low-price markets are opening up
  - Result: Extremely wide performance range

Performance Trends (FP16 GFLOPS)



2010 SGX 535

2011 SGX 543MP2

2012 SGX 543MP3

2013 G6430

2014 Adreno, K1, GX6650

Common Mobile GPU Families

Qualcomm Snapdragon Adreno

Old: Adreno 2xx	Now: Adreno 3xx	Soon: Adreno 4xx

ARM Mali

Old: 400	Now: T604, T628	Soon: T720, T760

Imagination Technologies

Old: SGX 5xx	Now: Series 6	Soon: Series 6XT

NVIDIA Tegra

Old: Tegra 3, 4	Now: K1	Soon: ...

Tile-based Mobile GPU

- Mobile GPUs are usually tile-based (next-gen too)

  Tile-based: ImgTec, Qualcomm*, ARM

  Direct: NVIDIA, Intel, Qualcomm*, Vivante

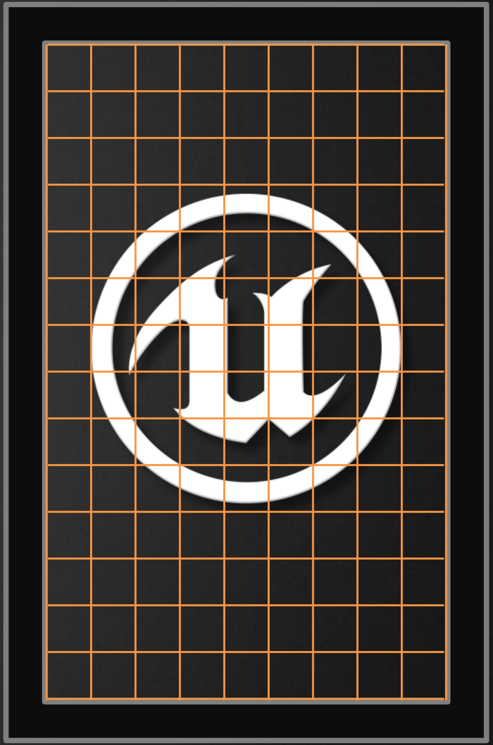
* Qualcomm Adreno can render either tile-based or direct to frame buffer

  – Extension: GL_QCOM_binning_control

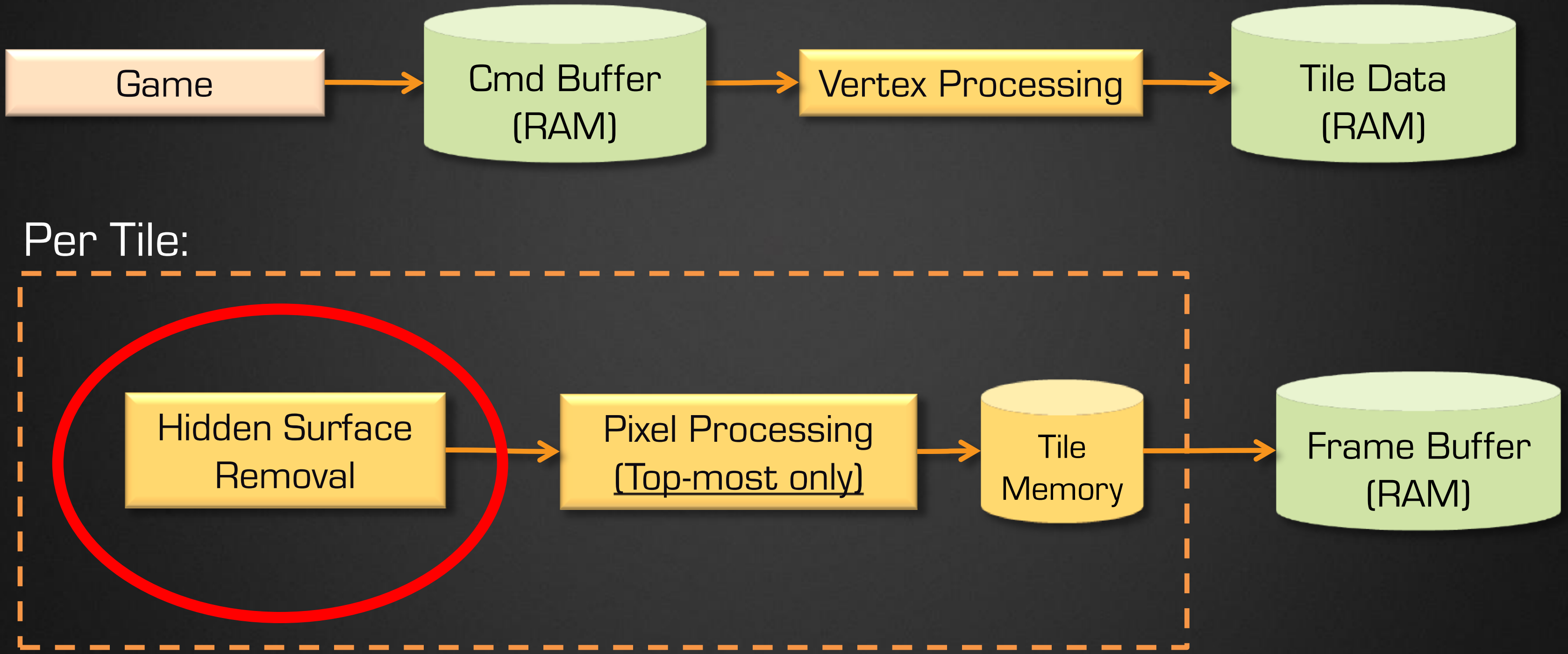
Tile-Based Mobile GPU

Summary:

- Split the screen into tiles
  - E.g. 32x32 pixels (ImgTec) or 300x300 (Qualcomm)
- The whole tile fits within GPU, on chip
- Process all drawcalls for one tile
  - Write out final tile results to RAM
- Repeat for each tile to fill the image in RAM



ImgTec Tile-based Rendering Process

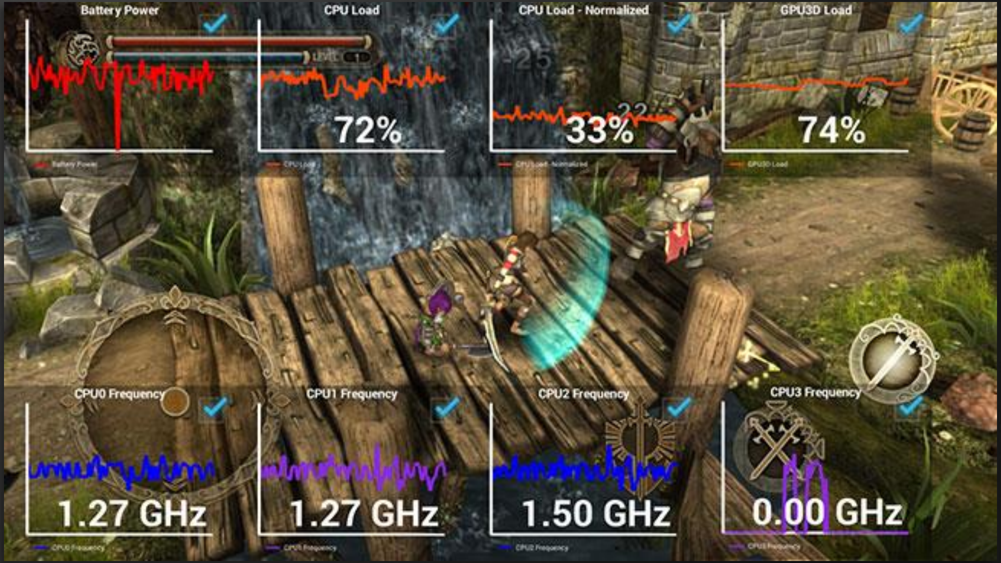


Framebuffer Resolve/Restore

- Expensive to switch Frame Buffer Object on Tile-based GPUs
  - Saves the current FBO to RAM
  - Reloads the new FBO from RAM
- Best performance:
  - A single rendertarget for the entire frame
  - No post-processing passes
- Does not apply to NVIDIA Tegra GPUs!
  - This made it simpler for us to use our full desktop rendering pipeline on K1
  - *"Rivalry"* tech demo (showing 5:00 pm today)

Thermal Limits

- Hardware CPU and GPU clock frequencies change all the time!
  - Many times per milli-second!
  - To save battery
  - To prevent overheating
- Qualcomm Trepn Profiler
  - https://developer.qualcomm.com/mobile-development/increase-app-performance/trepn-profiler



Thermal Limits

- Check your performance when device is cool
- Check again when it’s hot
- CPU uses much more power and heat than the GPU
    - Also, memory bandwidth generates a lot of heat
- Avoid unnecessary CPU usage
    - Spin-loops
    - Frequently waking up threads just to put them to sleep again

Performance Guidelines

- Always make sure lighting has been built before looking at performance
- Use as little post-process effects as you can get away with
- Make sure precomputed visibility has been set up properly
- Minimize overdraw (translucent or masked materials)
- Target 100-700 draw calls per frame
- Use as few texture lookups as possible in your materials
- Documentation:
  - https://docs.unrealengine.com/latest/INT/Platforms/Mobile/Performance/index.html

Performance Tier 1 – 2

1. 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

2. 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 Tier 3 – 4

3. 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

4. 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

Interlude: End of Part 1

Questions?

Keep going?

Coffee break?

Ready for more?

Part 2: Adapt and Conquer

- Very difficult to scale on CPU performance
  - Gameplay features can't easy be switched off
  - Also, CPUs aren't as different as GPUs are
  - Make sure you are never gamethread-bound on any device

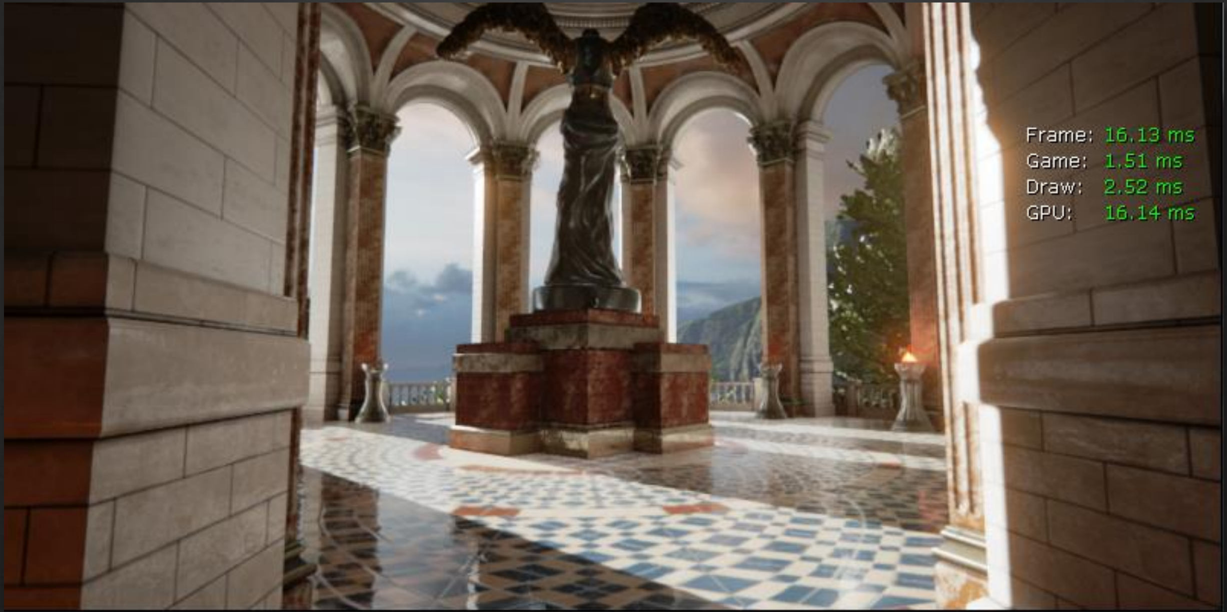
- Scale your game purely based on GPU performance
  - Primarily resolution and post-process effects
  - Ship it!

Cross-platform Console Commands

- Common commands:
  - Stat Unit
  - Stat UnitGraph
  - Stat FPS
  - Stat SceneRendering
  - Stat Slow
  - ViewMode ShaderComplexity
- Documentation:
  - https://docs.unrealengine.com/latest/INT/Engine/Rendering/PerformanceProfiling/StatCommands/index.html

Console Command: Stat Unit

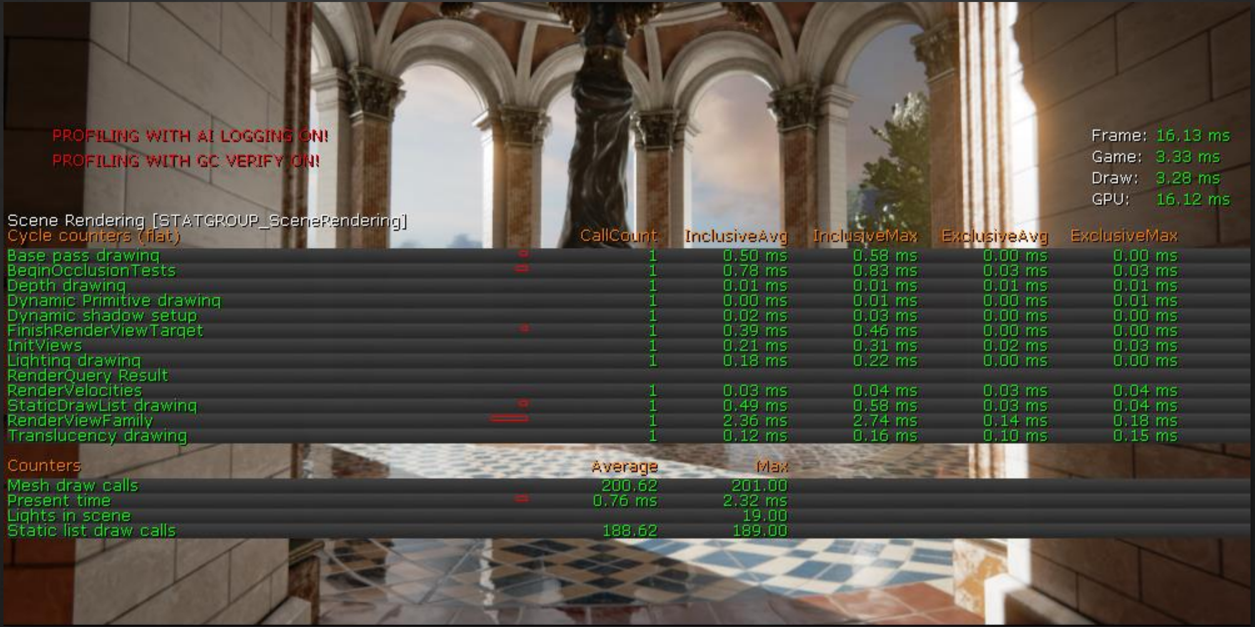
- Always the first step when checking performance



```
Frame:  16.13 ms
Game:   1.51 ms
Draw:   2.52 ms
GPU:    16.14 ms
```

Console Command: Stat SceneRendering

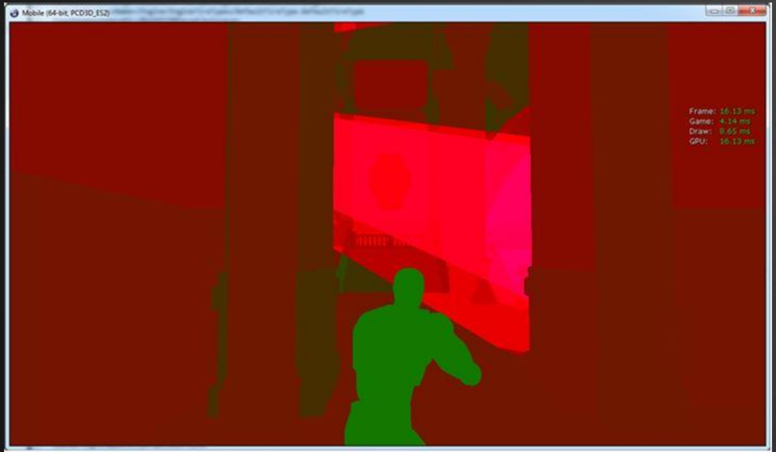
- Shows Renderthread CPU performance and drawcalls



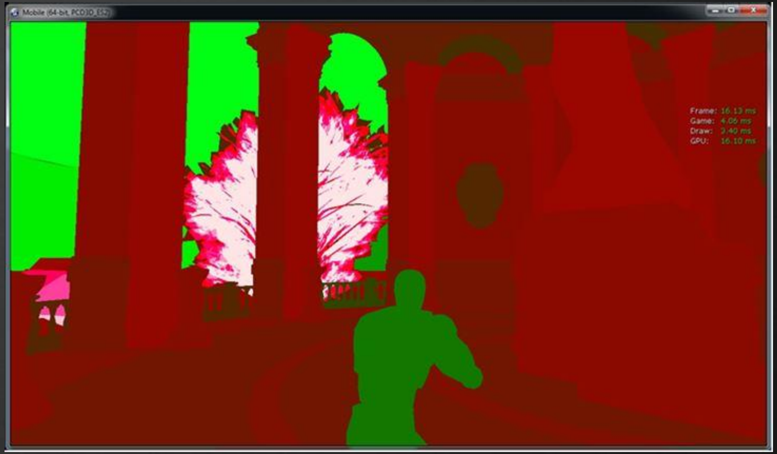
```
PROFILING WITH AI LOGGING ON!                                                       Frame:  16.13 ms
PROFILING WITH GC VERIFY ON!                                                        Game:   3.33 ms
                                                                                    Draw:   3.28 ms
                                                                                    GPU:    16.12 ms
Scene Rendering [STATGROUP_SceneRendering]
Cycle counters (flat)                  CallCount  InclusiveAvg  InclusiveMax  ExclusiveAvg  ExclusiveMax
Base pass drawing                              1     0.50 ms       0.58 ms       0.00 ms       0.00 ms
BeginOcclusionTests                            1     0.78 ms       0.83 ms       0.03 ms       0.03 ms
Depth drawing                                  1     0.01 ms       0.01 ms       0.01 ms       0.01 ms
Dynamic Primitive drawing                      1     0.00 ms       0.01 ms       0.00 ms       0.01 ms
Dynamic shadow setup                           1     0.02 ms       0.03 ms       0.00 ms       0.00 ms
FinishRenderViewTarget                         1     0.39 ms       0.46 ms       0.00 ms       0.00 ms
InitViews                                      1     0.21 ms       0.31 ms       0.02 ms       0.03 ms
Lighting drawing                               1     0.18 ms       0.22 ms       0.00 ms       0.00 ms
RenderQuery Result
RenderVelocities                               1     0.03 ms       0.04 ms       0.03 ms       0.04 ms
StaticDrawList drawing                         1     0.49 ms       0.58 ms       0.03 ms       0.04 ms
RenderViewFamily                               1     2.36 ms       2.74 ms       0.14 ms       0.18 ms
Translucency drawing                           1     0.12 ms       0.16 ms       0.10 ms       0.15 ms

Counters                                 Average         Max
Mesh draw calls                           200.62      201.00
Present time                             0.76 ms     2.32 ms
Lights in scene                                        19.00
Static list draw calls                    188.62      189.00
```

Console Commmand: ViewMode ShaderComplexity

- Visualize expensive materials in the PC ES2 previewer
- Shows approximate performance cost per material
- Green is good, red is bad. Pink or white is extremely expensive!



```
Frame: 16.13 ms
Game:  4.14 ms
Draw:  0.65 ms
GPU:   16.13 ms
```



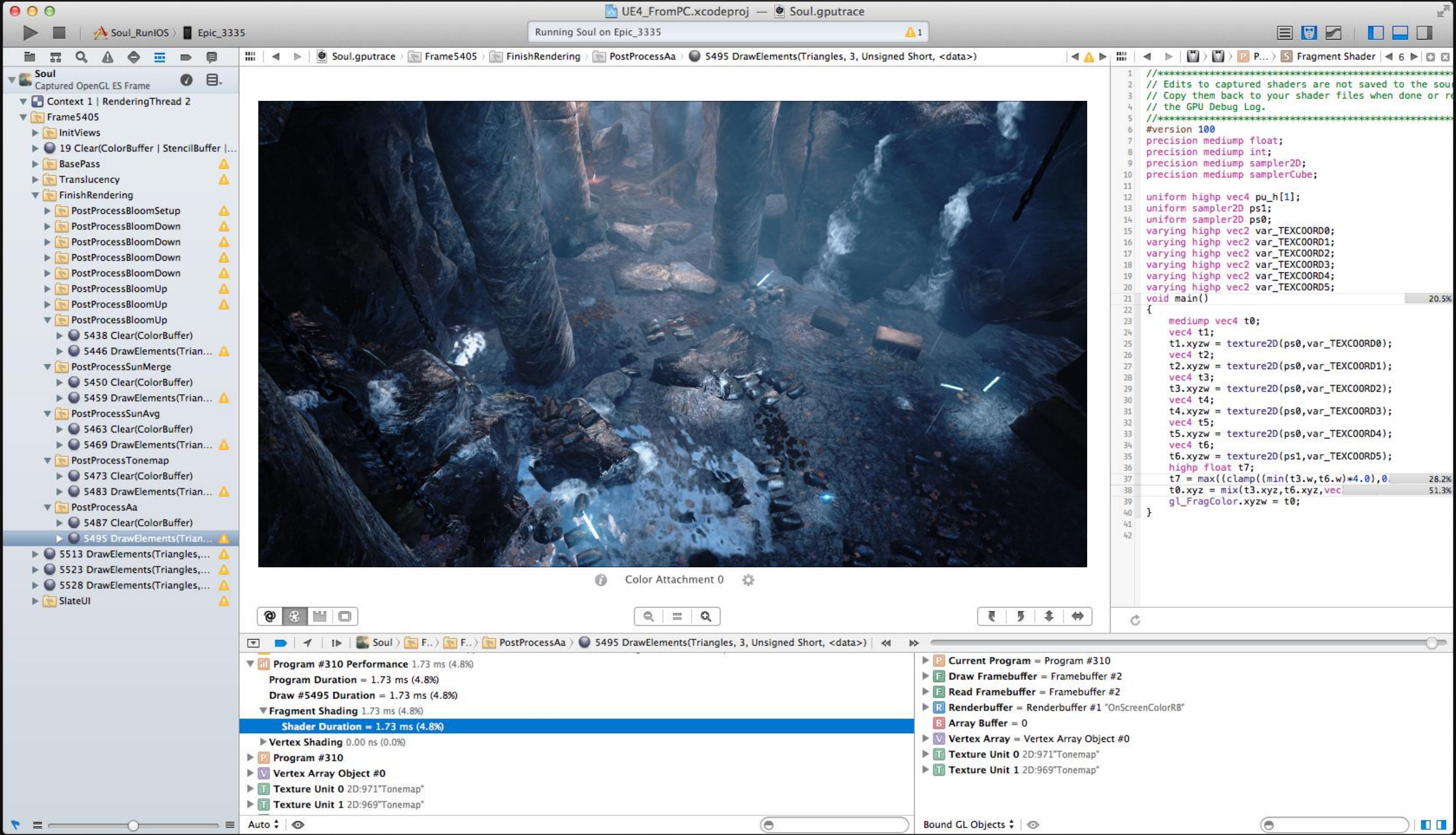
```
Frame: 16.13 ms
Game:  4.06 ms
Draw:  3.40 ms
GPU:   16.10 ms
```

iOS Performance

- New Metal graphics API in iOS 8
  - Much faster on CPU
  - Up to 20x faster on renderthread
  - Allows for thousands of drawcalls on iOS devices with A7 processors
- Scale graphics quality based on exact device model
  - Still very few different device models, easy to target each one specifically
  - Resolution (MobileContentScaleFactor)
  - Post-process features
  - Etc…

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
- For CPU profiling
  - Apple: Instruments (Time Profiler)
  - NVIDIA: Tegra System Profiler
  - ARM: DS-5



UE4_FromPC.xcodeproj — Soul.gputrace

Soul_RunIOS 〉 Epic_3335

Running Soul on Epic_3335 ⚠1

Soul.gputrace 〉 Frame5405 〉 FinishRendering 〉 PostProcessAa 〉 5495 DrawElements(Triangles, 3, Unsigned Short, <data>)

P... 〉 Fragment Shader ◀ 6 ▶

Soul
Captured OpenGL ES Frame

- Context 1 | RenderingThread 2
  - Frame5405
    - InitViews
    - 19 Clear(ColorBuffer | StencilBuffer |...
    - BasePass ⚠
    - Translucency ⚠
    - FinishRendering
      - PostProcessBloomSetup ⚠
      - PostProcessBloomDown ⚠
      - PostProcessBloomDown ⚠
      - PostProcessBloomDown ⚠
      - PostProcessBloomDown ⚠
      - PostProcessBloomUp ⚠
      - PostProcessBloomUp ⚠
      - PostProcessBloomUp
        - 5438 Clear(ColorBuffer)
        - 5446 DrawElements(Trian... ⚠
      - PostProcessSunMerge
        - 5450 Clear(ColorBuffer)
        - 5459 DrawElements(Trian... ⚠
      - PostProcessSunAvg
        - 5463 Clear(ColorBuffer)
        - 5469 DrawElements(Trian... ⚠
      - PostProcessTonemap
        - 5473 Clear(ColorBuffer)
        - 5483 DrawElements(Trian... ⚠
      - PostProcessAa
        - 5487 Clear(ColorBuffer)
        - 5495 DrawElements(Trian... ⚠
    - 5513 DrawElements(Triangles,... ⚠
    - 5523 DrawElements(Triangles,... ⚠
    - 5528 DrawElements(Triangles,... ⚠
    - SlateUI ⚠

```
//************************************************************
// Edits to captured shaders are not saved to the sou
// Copy them back to your shader files when done or re
// the GPU Debug Log.
//************************************************************
#version 100
precision mediump float;
precision mediump int;
precision mediump sampler2D;
precision mediump samplerCube;

uniform highp vec4 pu_h[1];
uniform sampler2D ps1;
uniform sampler2D ps0;
varying highp vec2 var_TEXCOORD0;
varying highp vec2 var_TEXCOORD1;
varying highp vec2 var_TEXCOORD2;
varying highp vec2 var_TEXCOORD3;
varying highp vec2 var_TEXCOORD4;
varying highp vec2 var_TEXCOORD5;
void main()                                          20.5%
{
    mediump vec4 t0;
    vec4 t1;
    t1.xyzw = texture2D(ps0,var_TEXCOORD0);
    vec4 t2;
    t2.xyzw = texture2D(ps0,var_TEXCOORD1);
    vec4 t3;
    t3.xyzw = texture2D(ps0,var_TEXCOORD2);
    vec4 t4;
    t4.xyzw = texture2D(ps0,var_TEXCOORD3);
    vec4 t5;
    t5.xyzw = texture2D(ps0,var_TEXCOORD4);
    vec4 t6;
    t6.xyzw = texture2D(ps1,var_TEXCOORD5);
    highp float t7;
    t7 = max((clamp((min(t3.w,t6.w)*4.0),0.        28.2%
    t0.xyz = mix(t3.xyz,t6.xyz,vec                 51.3%
    gl_FragColor.xyzw = t0;
}
```

Color Attachment 0

Soul 〉 F.. 〉 F.. 〉 PostProcessAa 〉 5495 DrawElements(Triangles, 3, Unsigned Short, <data>)

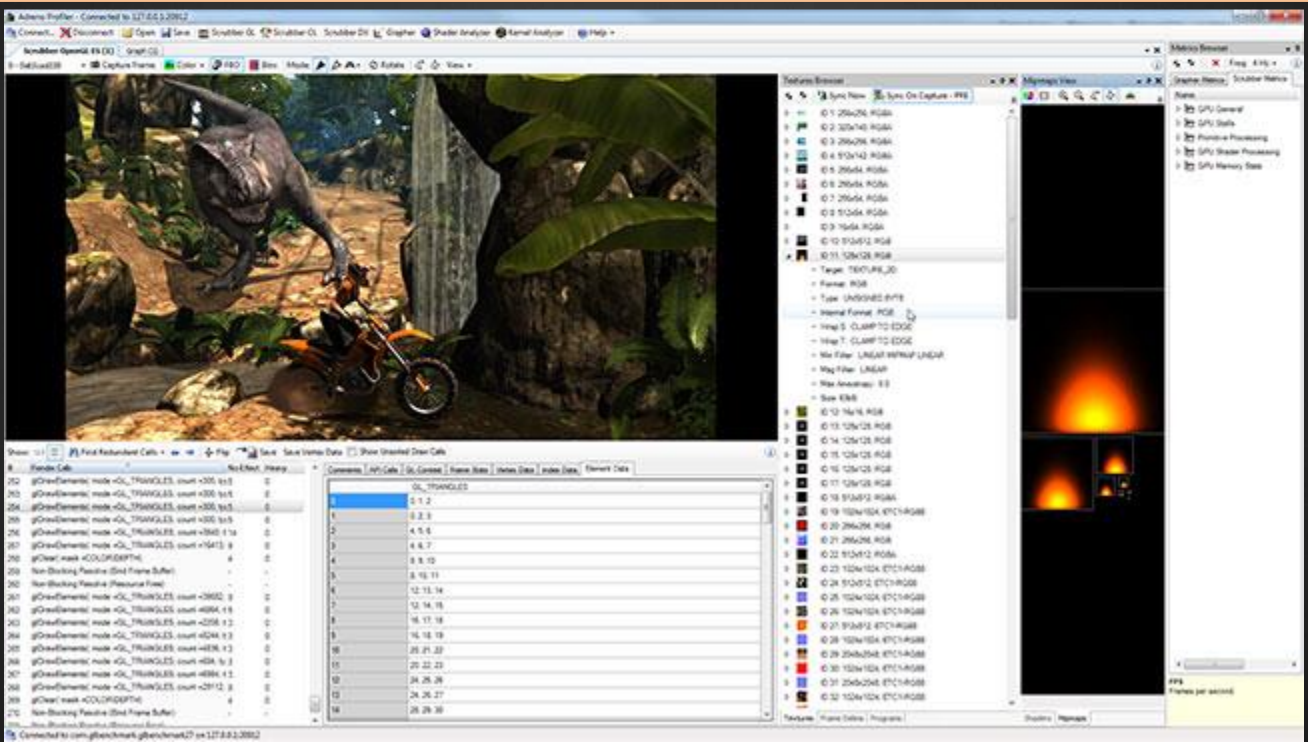
- **Program #310 Performance** 1.73 ms (4.8%)
  - **Program Duration** = 1.73 ms (4.8%)
  - **Draw #5495 Duration** = 1.73 ms (4.8%)
  - **Fragment Shading** 1.73 ms (4.8%)
    - **Shader Duration** = 1.73 ms (4.8%)
  - **Vertex Shading** 0.00 ns (0.0%)
- **Program #310**
- **Vertex Array Object #0**
- **Texture Unit 0** 2D:971"Tonemap"
- **Texture Unit 1** 2D:969"Tonemap"

Auto

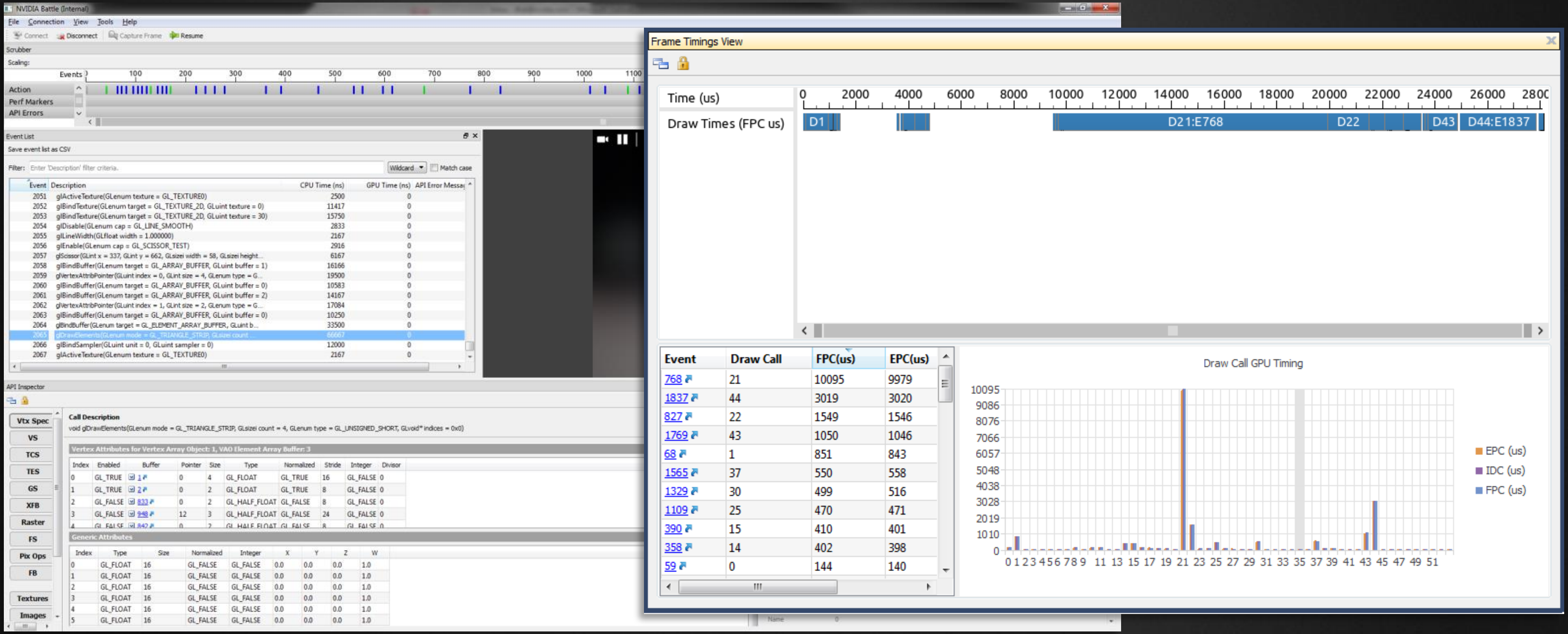
- **Current Program** = Program #310
- **Draw Framebuffer** = Framebuffer #2
- **Read Framebuffer** = Framebuffer #2
- **Renderbuffer** = Renderbuffer #1 "OnScreenColorRB"
- **Array Buffer** = 0
- **Vertex Array** = Vertex Array Object #0
- **Texture Unit 0** 2D:971"Tonemap"
- **Texture Unit 1** 2D:969"Tonemap"

Bound GL Objects

Qualcomm Adreno Profiler



NVIDIA Tegra Graphics Debugger



Event	Description	CPU Time (ns)	GPU Time (ns)	API Error Messa
2051	glActiveTexture(GLenum texture = GL_TEXTURE0)	2500	0	
2052	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 0)	11417	0	
2053	glBindTexture(GLenum target = GL_TEXTURE_2D, GLuint texture = 30)	15750	0	
2054	glDisable(GLenum cap = GL_LINE_SMOOTH)	2833	0	
2055	glLineWidth(GLfloat width = 1.000000)	2167	0	
2056	glEnable(GLenum cap = GL_SCISSOR_TEST)	2916	0	
2057	glScissor(GLint x = 337, GLint y = 662, GLsizei width = 58, GLsizei height...	6167	0	
2058	glBindBuffer(GLenum target = GL_ARRAY_BUFFER, GLuint buffer = 1)	16166	0	
2059	glVertexAttribPointer(GLuint index = 0, GLint size = 4, GLenum type = G...	19500	0	
2060	glBindBuffer(GLenum target = GL_ARRAY_BUFFER, GLuint buffer = 0)	10583	0	
2061	glBindBuffer(GLenum target = GL_ARRAY_BUFFER, GLuint buffer = 2)	14167	0	
2062	glVertexAttribPointer(GLuint index = 1, GLint size = 2, GLenum type = G...	17084	0	
2063	glBindBuffer(GLenum target = GL_ARRAY_BUFFER, GLuint buffer = 0)	10250	0	
2064	glBindBuffer(GLenum target = GL_ELEMENT_ARRAY_BUFFER, GLuint b...	33500	0	
2065	glDrawElements(GLenum mode = GL_TRIANGLE_STRIP, GLsizei count ...	66667	0	
2066	glBindSampler(GLuint unit = 0, GLuint sampler = 0)	12000	0	
2067	glActiveTexture(GLenum texture = GL_TEXTURE0)	2167	0	

Call Description

void glDrawElements(GLenum mode = GL_TRIANGLE_STRIP, GLsizei count = 4, GLenum type = GL_UNSIGNED_SHORT, GLvoid* indices = 0x0)

Vertex Attributes for Vertex Array Object: 1, VAO Element Array Buffer: 3

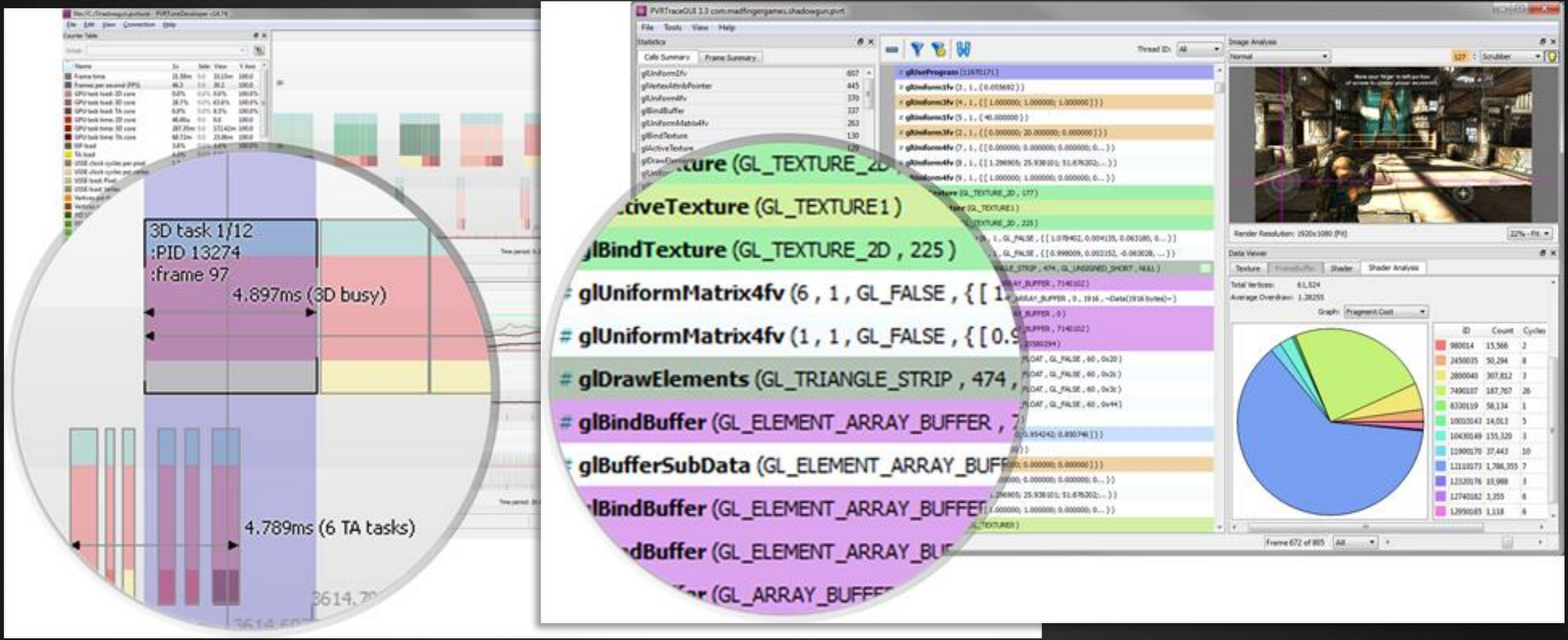
Index	Enabled	Buffer	Pointer	Size	Type	Normalized	Stride	Integer	Divisor
0	GL_TRUE	1	0	4	GL_FLOAT	GL_TRUE	16	GL_FALSE	0
1	GL_TRUE	2	0	2	GL_FLOAT	GL_TRUE	8	GL_FALSE	0
2	GL_FALSE	833	0	2	GL_HALF_FLOAT	GL_FALSE	8	GL_FALSE	0
3	GL_FALSE	948	12	3	GL_HALF_FLOAT	GL_FALSE	24	GL_FALSE	0
4	GL_FALSE	842	0	2	GL_HALF_FLOAT	GL_FALSE	8	GL_FALSE	0

Generic Attributes

Index	Type	Size	Normalized	Integer	X	Y	Z	W
0	GL_FLOAT	16	GL_FALSE	GL_FALSE	0.0	0.0	0.0	1.0
1	GL_FLOAT	16	GL_FALSE	GL_FALSE	0.0	0.0	0.0	1.0
2	GL_FLOAT	16	GL_FALSE	GL_FALSE	0.0	0.0	0.0	1.0
3	GL_FLOAT	16	GL_FALSE	GL_FALSE	0.0	0.0	0.0	1.0
4	GL_FLOAT	16	GL_FALSE	GL_FALSE	0.0	0.0	0.0	1.0
5	GL_FLOAT	16	GL_FALSE	GL_FALSE	0.0	0.0	0.0	1.0

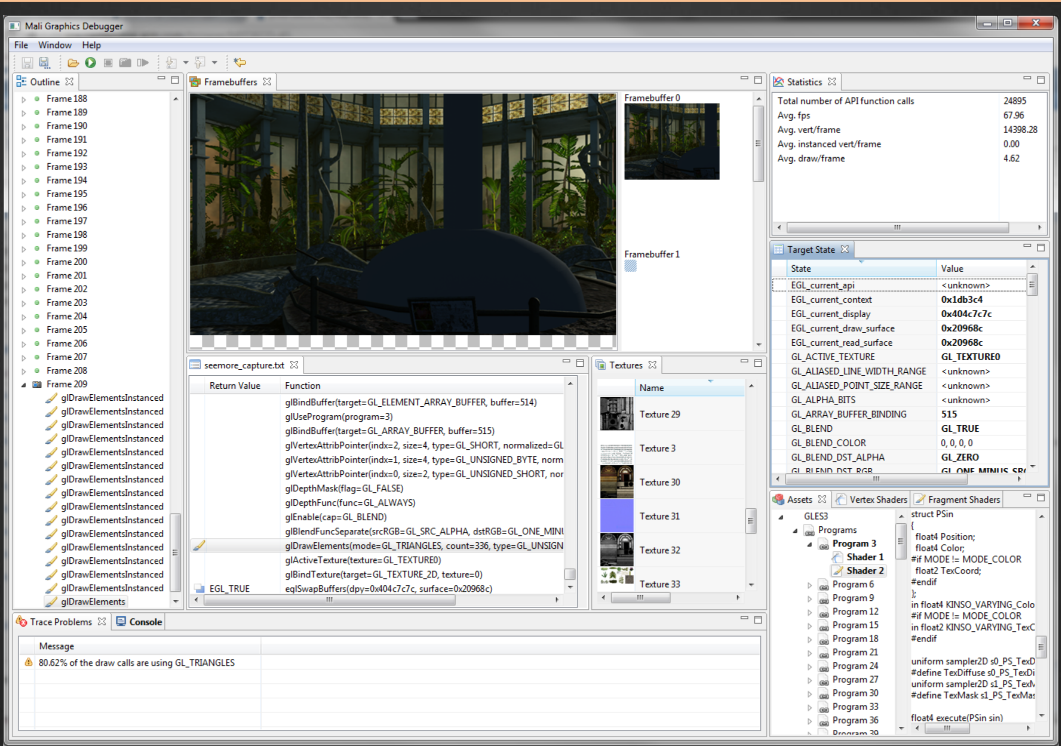
Event	Draw Call	FPC(us)	EPC(us)
768	21	10095	9979
1837	44	3019	3020
827	22	1549	1546
1769	43	1050	1046
68	1	851	843
1565	37	550	558
1329	30	499	516
1109	25	470	471
390	15	410	401
358	14	402	398
59	0	144	140

ImgTec PVRTune and PVRTrace



```
ture (GL_TEXTURE_2D
tiveTexture (GL_TEXTURE1)
glBindTexture (GL_TEXTURE_2D, 225)
# glUniformMatrix4fv (6, 1, GL_FALSE, {[ 1
# glUniformMatrix4fv (1, 1, GL_FALSE, {[ 0.9
# glDrawElements (GL_TRIANGLE_STRIP, 474,
# glBindBuffer (GL_ELEMENT_ARRAY_BUFFER, 7
# glBufferSubData (GL_ELEMENT_ARRAY_BUFF
lBindBuffer (GL_ELEMENT_ARRAY_BUFFER
dBuffer (GL_ELEMENT_ARRAY_BU
r (GL_ARRAY_BUFFE
```

ARM Mali Graphics Debugger



Frame 188
Frame 189
Frame 190
Frame 191
Frame 192
Frame 193
Frame 194
Frame 195
Frame 196
Frame 197
Frame 198
Frame 199
Frame 200
Frame 201
Frame 202
Frame 203
Frame 204
Frame 205
Frame 206
Frame 207
Frame 208
Frame 209
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElementsInstanced
glDrawElements

Return Value	Function
	glBindBuffer(target=GL_ELEMENT_ARRAY_BUFFER, buffer=514)
	glUseProgram(program=3)
	glBindBuffer(target=GL_ARRAY_BUFFER, buffer=515)
	glVertexAttribPointer(indx=2, size=4, type=GL_SHORT, normalized=GL
	glVertexAttribPointer(indx=1, size=4, type=GL_UNSIGNED_BYTE, norm
	glVertexAttribPointer(indx=0, size=2, type=GL_UNSIGNED_SHORT, nor
	glDepthMask(flag=GL_FALSE)
	glDepthFunc(func=GL_ALWAYS)
	glEnable(cap=GL_BLEND)
	glBlendFuncSeparate(srcRGB=GL_SRC_ALPHA, dstRGB=GL_ONE_MINU
	glDrawElements(mode=GL_TRIANGLES, count=336, type=GL_UNSIGN
	glActiveTexture(texture=GL_TEXTURE0)
	glBindTexture(target=GL_TEXTURE_2D, texture=0)
EGL_TRUE	eglSwapBuffers(dpy=0x404c7c7c, surface=0x20968c)

Name
Texture 29
Texture 3
Texture 30
Texture 31
Texture 32
Texture 33

Total number of API function calls	24895
Avg. fps	67.96
Avg. vert/frame	14398.28
Avg. instanced vert/frame	0.00
Avg. draw/frame	4.62

State	Value
EGL_current_api	<unknown>
EGL_current_context	**0x1db3c4**
EGL_current_display	**0x404c7c7c**
EGL_current_draw_surface	**0x20968c**
EGL_current_read_surface	**0x20968c**
GL_ACTIVE_TEXTURE	**GL_TEXTURE0**
GL_ALIASED_LINE_WIDTH_RANGE	<unknown>
GL_ALIASED_POINT_SIZE_RANGE	<unknown>
GL_ALPHA_BITS	<unknown>
GL_ARRAY_BUFFER_BINDING	**515**
GL_BLEND	**GL_TRUE**
GL_BLEND_COLOR	0, 0, 0, 0
GL_BLEND_DST_ALPHA	**GL_ZERO**
GL_BLEND_DST_RGB	**GL_ONE_MINUS_SR**

GLES3
Programs
Program 3
Shader 1
Shader 2
Program 6
Program 9
Program 12
Program 15
Program 18
Program 21
Program 24
Program 27
Program 30
Program 33
Program 36
Program 39

```
struct PSin
{
 float4 Position;
 float4 Color;
#if MODE != MODE_COLOR
 float2 TexCoord;
#endif
};
in float4 KINSO_VARYING_Colo
#if MODE != MODE_COLOR
in float2 KINSO_VARYING_TexC
#endif

uniform sampler2D s0_PS_TexD
#define TexDiffuse s0_PS_TexDi
uniform sampler2D s1_PS_TexM
#define TexMask s1_PS_TexMas

float4 execute(PSin sin)
```

Message
⚠ 80.62% of the draw calls are using GL_TRIANGLES

Device Profiles

- UE4 selects one device profile at startup
  - Detects device model and capabilities
- Tweak each device profile for your game
  - Config/DefaultDeviceProfiles.ini
  - Each Device Profile can customize engine features, like:
    - +CVars=r.MobileContentScaleFactor=2
    - +CVars=r.BloomQuality=1
    - +CVars=r.DepthOfFieldQuality=1
    - +CVars=r.LightShaftQuality=1
- Documentation:
  - https://docs.unrealengine.com/latest/INT/Platforms/DeviceProfiles/index.html

UE4 Mobile Performance Questions?

Documentation, Tutorials and Help at:

AnswerHub:	http://answers.unrealengine.com
Engine Documentation:	http://docs.unrealengine.com
Official Forums:	http://forums.unrealengine.com
Community Wiki:	http://wiki.unrealengine.com
YouTube Videos:	http://www.youtube.com/user/UnrealDevelopmentKit
Community IRC:	#unrealengine on FreeNode

Unreal Engine 4 Roadmap

- lmgtfy.com/?q=Unreal+engine+Trello+