Android Game Optimization
Deep Dive

Jonas Gustavsson & Jungwoo Kim
Samsung Electronics

Android Game Developer Summit 2017

Introduction

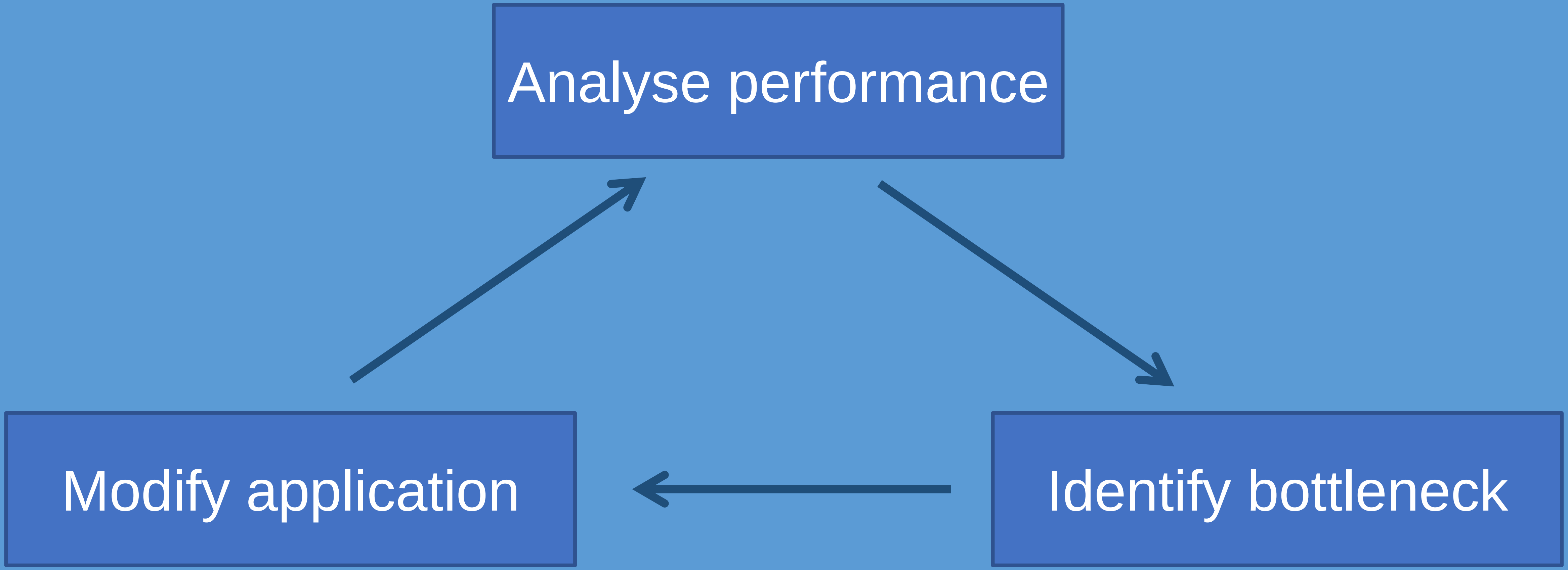
- Look inside of performance Tools
- Vulkan Optimization case studies

Look inside of performance Tools

Jonas Gustavsson

Profiling

Performance Analysis Workflow



Analyse performance

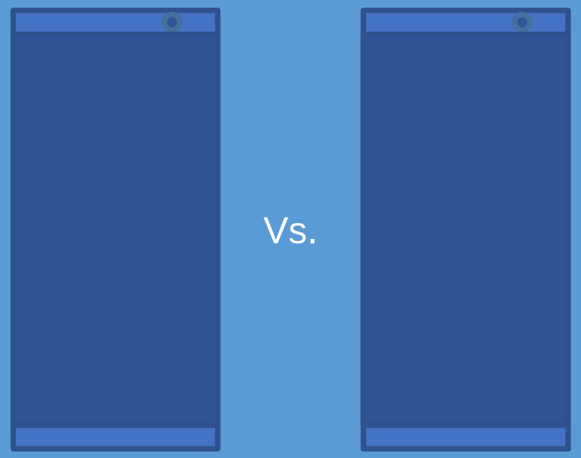
Identify bottleneck

Modify application

Understanding your device

- Lots of variation between devices
  - Some are obvious, e.g. screen resolution & GPU model
  - Some are subtle, e.g. memory bus speed

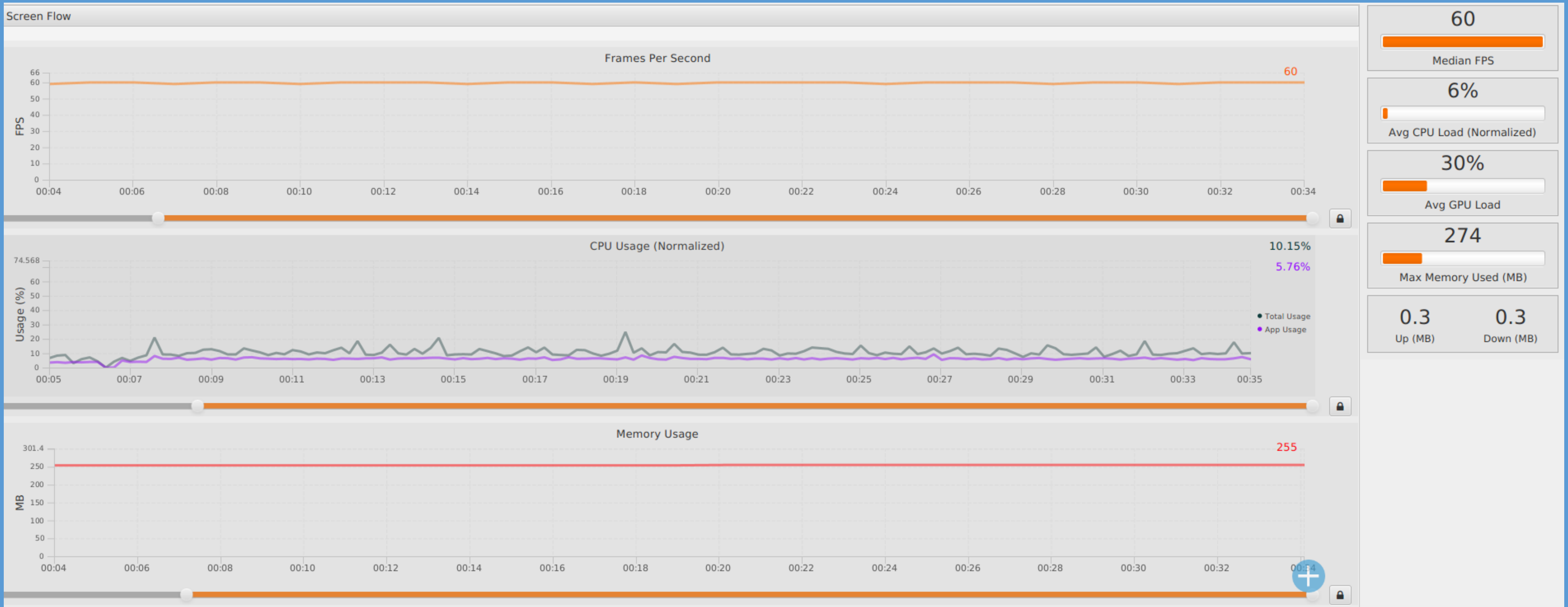
- Profiling **all** devices that matter to you is vital
  - We recommend mixing local and remote device testing



CPU & GPU performance analysis

Tool	Vendor	CPU	GPU
Simpleperf	Google	Y	N
Systrace	Google	Y	N
GameBench Desktop App	GameBench	Y	Y
DS-5 Streamline	ARM	Y	Y
Snapdragon Profiler	Qualcomm	Y	Y
Trepn Profiler (Android app)	Qualcomm	Y	Y
PVRTune	Imagination	Y	Y
Tegra Nsight	NVIDIA	Y	Y

GameBench Desktop App



60
Median FPS

6%
Avg CPU Load (Normalized)

30%
Avg GPU Load

274
Max Memory Used (MB)

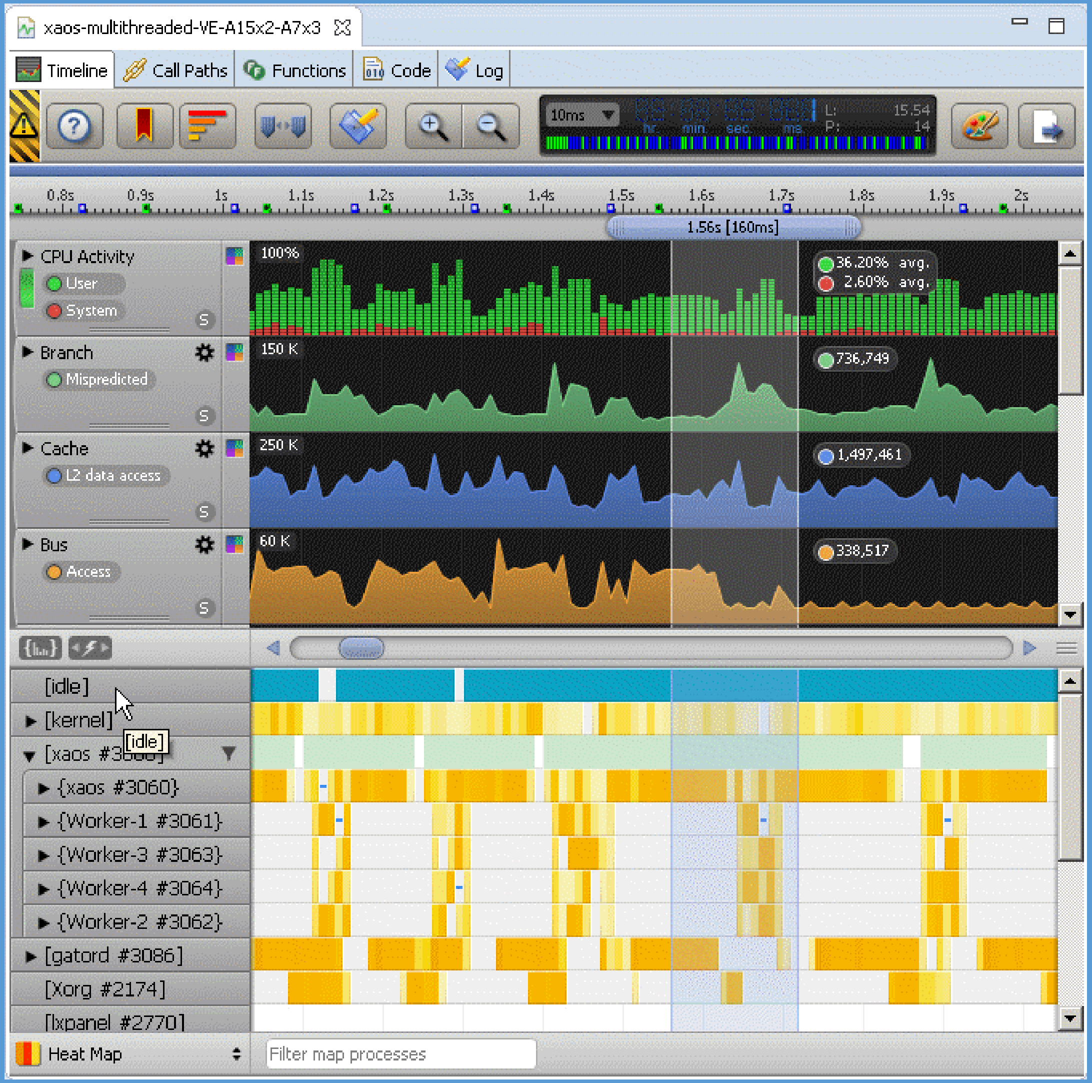
0.3
Up (MB)

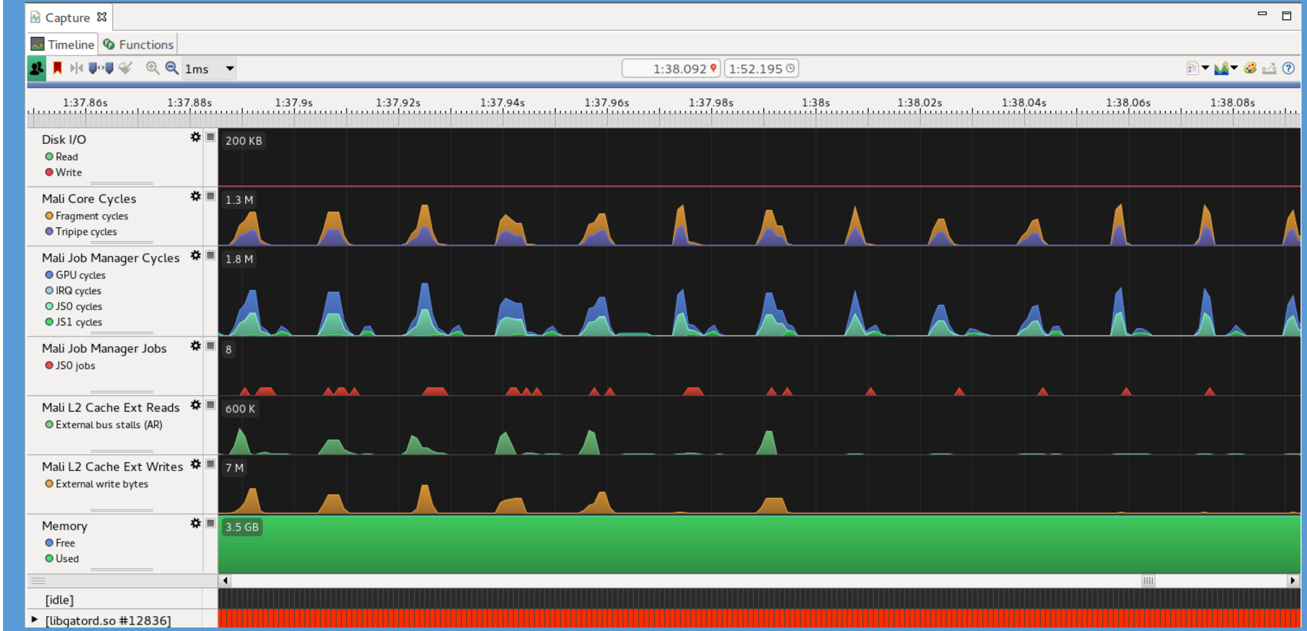
0.3
Down (MB)

ARM DS-5 Streamline

- ARM’s profiler
- Community Edition
  - Basic CPU & system counters
  - All Mali GPU counters
  - CE is free. Paid Editions for enhanced functionality

ARM – DS-5 Streamline



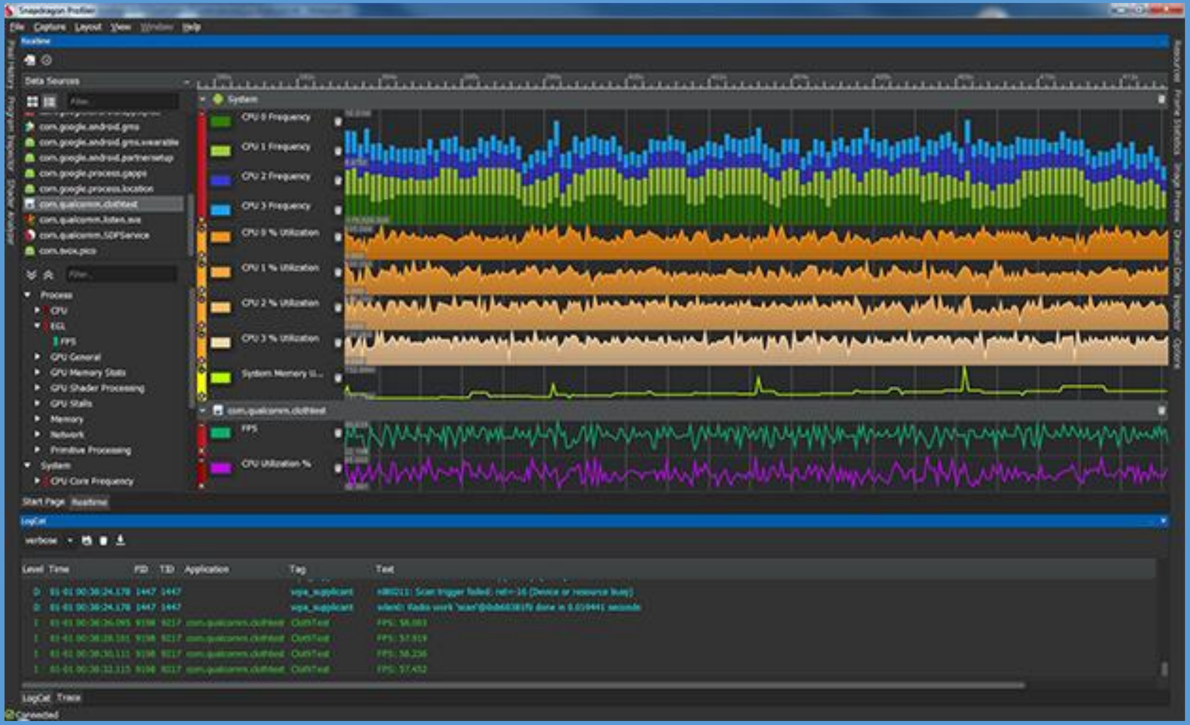


SAMSUNG

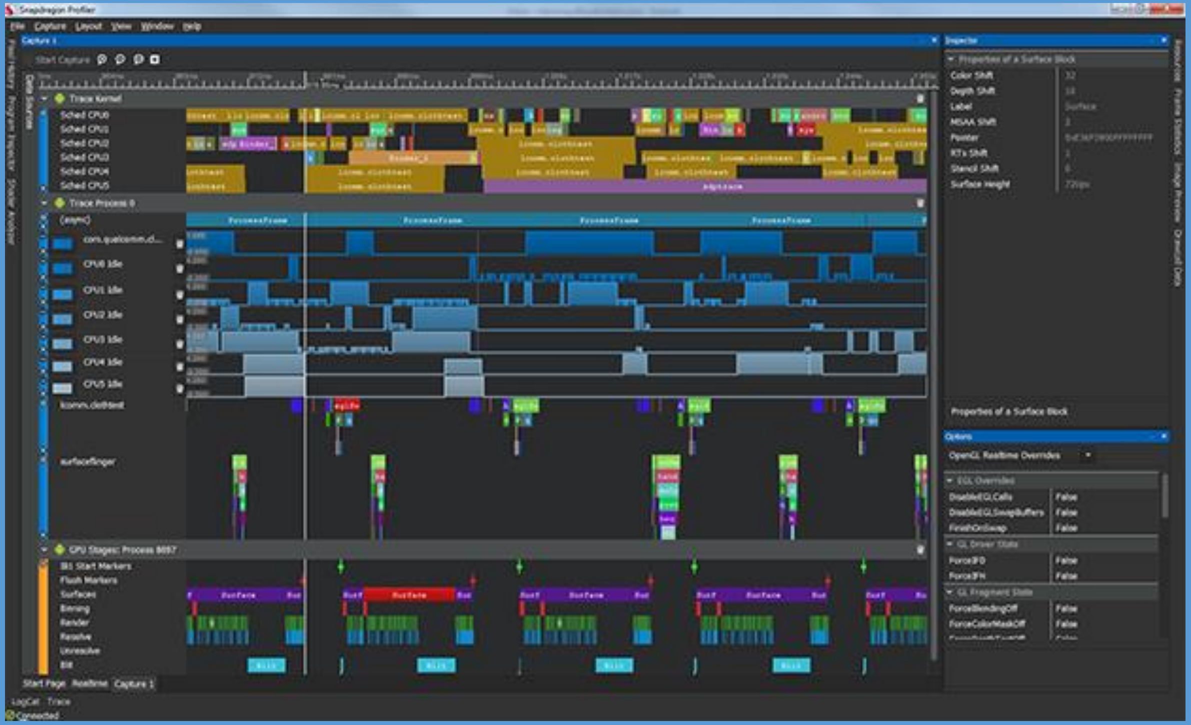
Qualcomm Snapdragon Profiler

- Qualcomm’s profiler
  - Analyze CPU, GPU, DSP, memory, power, thermal, and network data

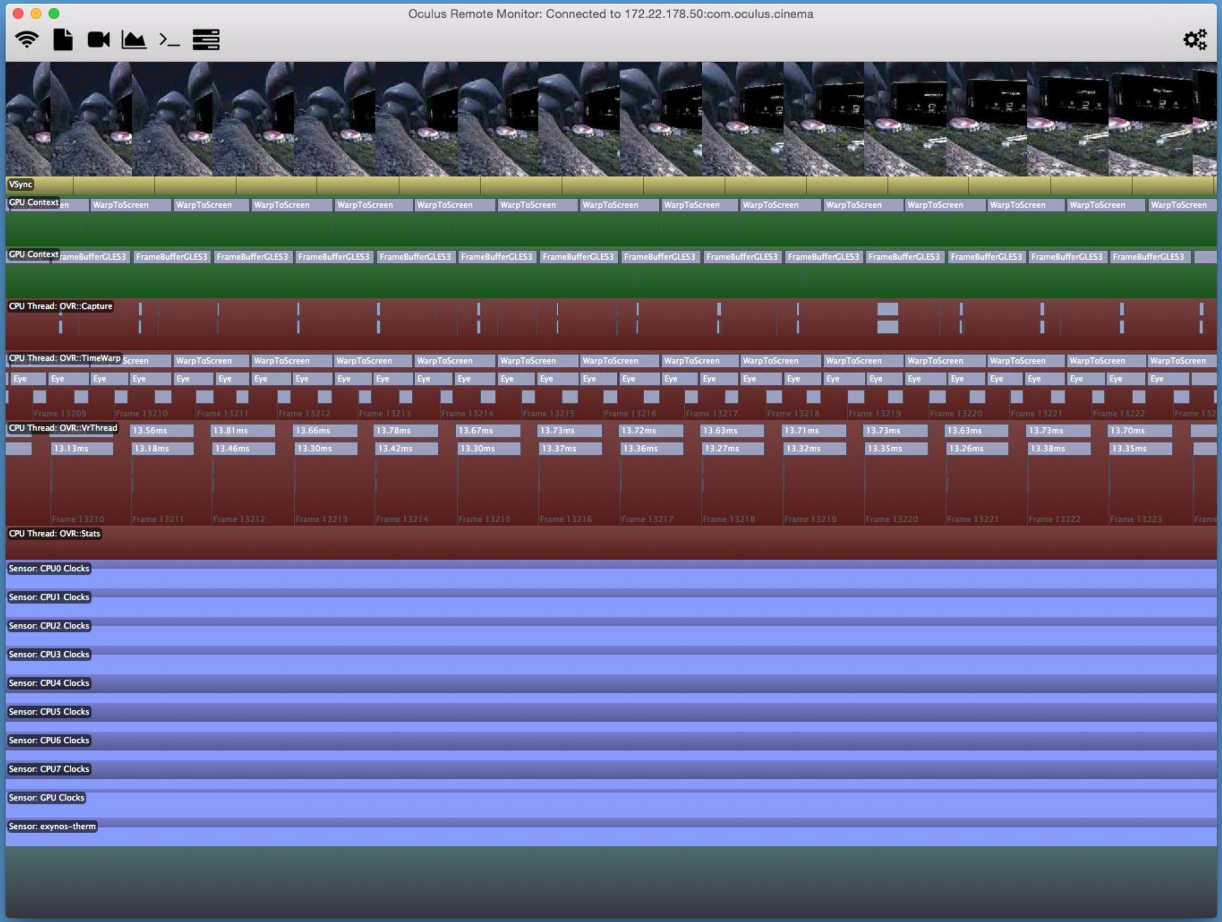
Qualcomm Snapdragon Profiler



```
Level Time               PID  TID  Application             Tag             Text
D     01-01 00:38:24.178 1447 1447                         wpa_supplicant  nl80211: Scan trigger failed: ret=-16 (Device or resource busy)
D     01-01 00:38:24.178 1447 1447                         wpa_supplicant  wlan0: Radio work 'scan'@0xb6838 1f8 done in 0.019441 seconds
I     01-01 00:38:26.095 9198 9217 com.qualcomm.clothtest  ClothTest       FPS: 58.003
I     01-01 00:38:28.101 9198 9217 com.qualcomm.clothtest  ClothTest       FPS: 57.919
I     01-01 00:38:30.111 9198 9217 com.qualcomm.clothtest  ClothTest       FPS: 58.236
I     01-01 00:38:32.115 9198 9217 com.qualcomm.clothtest  ClothTest       FPS: 57.452
```



GearVR:
Oculus Performance Data Viewer

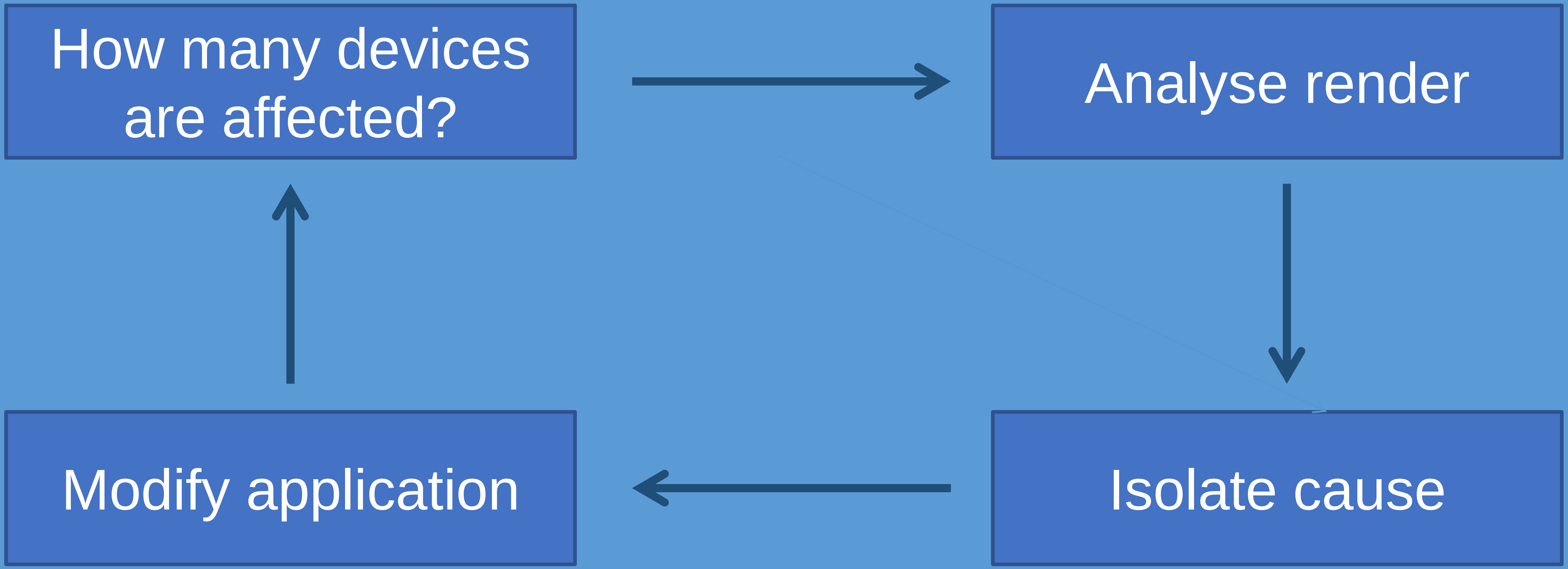


Thermal throttling vs. profiling

- Dynamic power management
  - Useful for power saving
  - Annoying for profiling!
- OEMs are beginning to support Android’s Sustained Performance API

Debugging OpenGL ES & Vulkan

Performance Analysis Workflow



How many devices are affected?

Analyse render

Isolate cause

Modify application

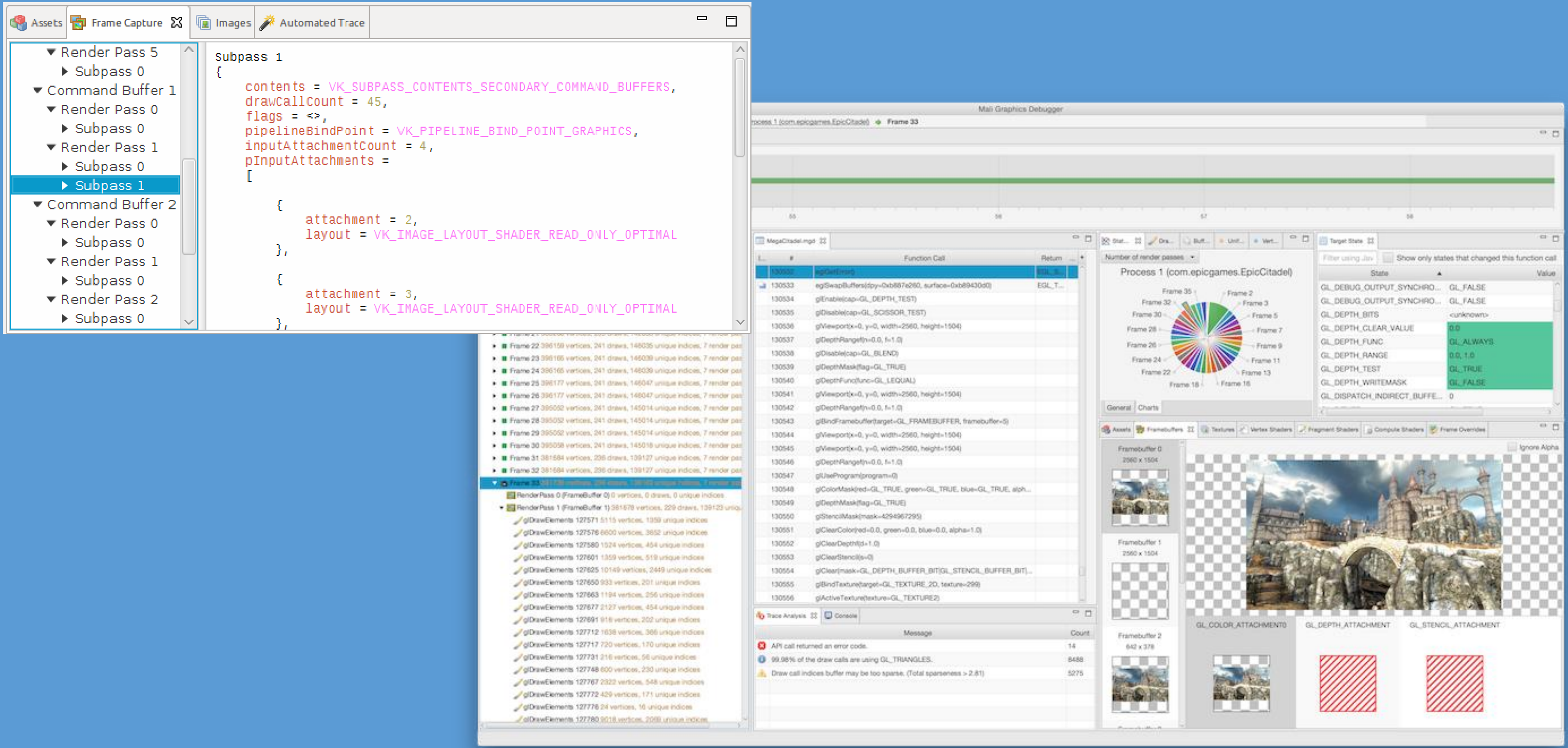
Graphics API capture & analysis

Tool	Vendor	OpenGL ES	Vulkan
GAPID	Google	Y	Y
Mali Graphics Debugger	ARM	Y	Y
Snapdragon Profiler	Qualcomm	Y	Y
PVRTrace	Imagination	Y	N
Tegra NSight	NVIDIA	Y	Y
vkTrace	LunarG	N	Y
RenderDoc	RenderDoc	Y (in progress)	Y (alpha quality)

API integrated tools

- OpenGL ES
  - KHR_debug/debug output
- Vulkan
  - Validation layers
    - Important to be error free before shipping!

ARM Mali Graphics Debugger



- Render Pass 5
  - Subpass 0
- Command Buffer 1
  - Render Pass 0
    - Subpass 0
  - Render Pass 1
    - Subpass 0
    - Subpass 1
- Command Buffer 2
  - Render Pass 0
    - Subpass 0
  - Render Pass 1
    - Subpass 0
  - Render Pass 2
    - Subpass 0

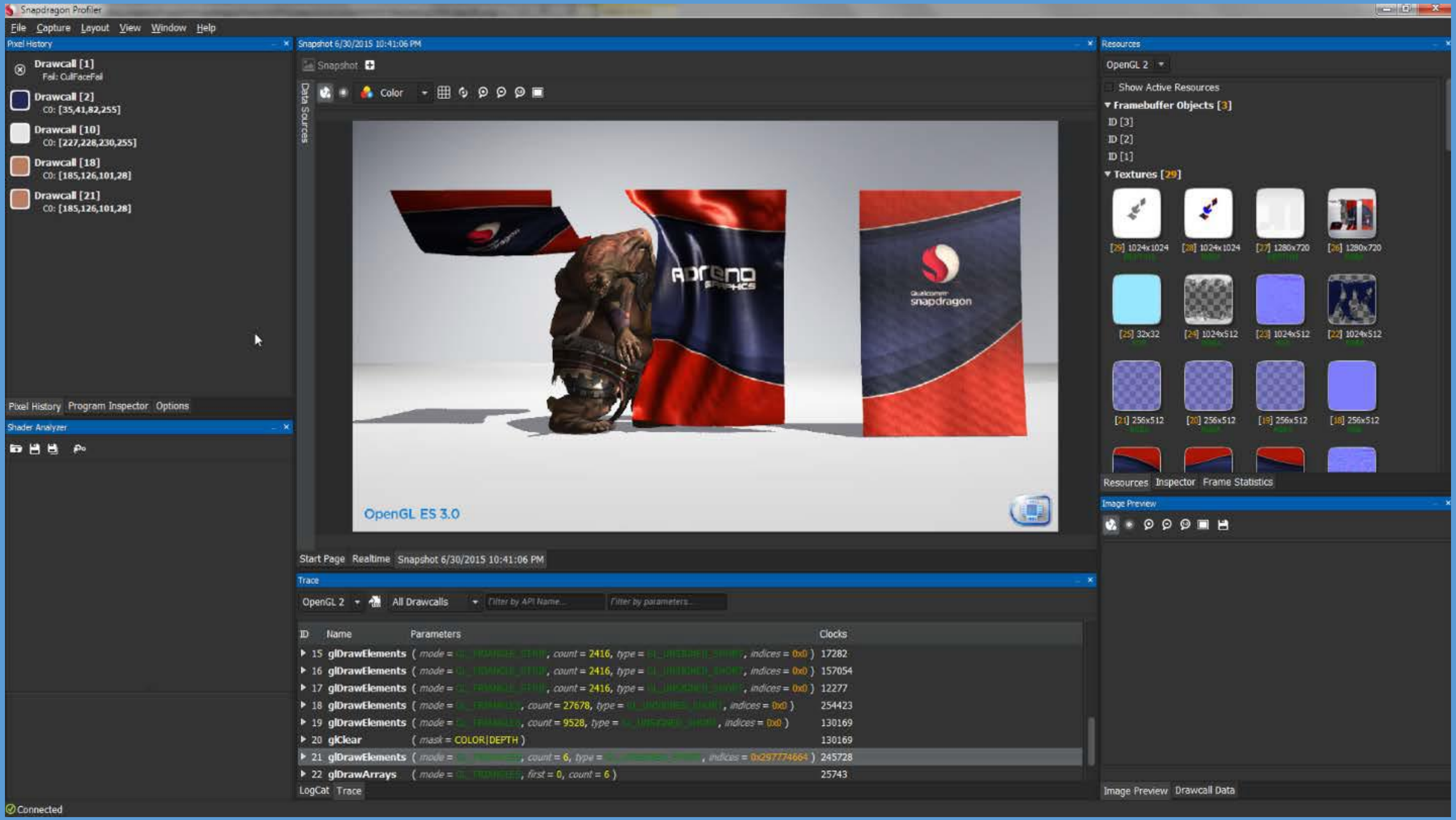
```
Subpass 1
{
    contents = VK_SUBPASS_CONTENTS_SECONDARY_COMMAND_BUFFERS,
    drawCallCount = 45,
    flags = <>,
    pipelineBindPoint = VK_PIPELINE_BIND_POINT_GRAPHICS,
    inputAttachmentCount = 4,
    pInputAttachments =
    [

        {
            attachment = 2,
            layout = VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL
        },

        {
            attachment = 3,
            layout = VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL
        },
```

#	Function Call	Return
130532	eglGetError()	EGL_S...
130533	eglSwapBuffers(dpy=0xb887e260, surface=0xb89430d0)	EGL_T...
130534	glEnable(cap=GL_DEPTH_TEST)	
130535	glDisable(cap=GL_SCISSOR_TEST)	
130536	glViewport(x=0, y=0, width=2560, height=1504)	
130537	glDepthRangef(n=0.0, f=1.0)	
130538	glDisable(cap=GL_BLEND)	
130539	glDepthMask(flag=GL_TRUE)	
130540	glDepthFunc(func=GL_LEQUAL)	
130541	glViewport(x=0, y=0, width=2560, height=1504)	
130542	glDepthRangef(n=0.0, f=1.0)	
130543	glBindFramebuffer(target=GL_FRAMEBUFFER, framebuffer=5)	
130544	glViewport(x=0, y=0, width=2560, height=1504)	
130545	glViewport(x=0, y=0, width=2560, height=1504)	
130546	glDepthRangef(n=0.0, f=1.0)	
130547	glUseProgram(program=0)	
130548	glColorMask(red=GL_TRUE, green=GL_TRUE, blue=GL_TRUE, alph...	
130549	glDepthMask(flag=GL_TRUE)	
130550	glStencilMask(mask=4294967295)	
130551	glClearColor(red=0.0, green=0.0, blue=0.0, alpha=1.0)	
130552	glClearDepthf(d=1.0)	
130553	glClearStencil(s=0)	
130554	glClear(mask=GL_DEPTH_BUFFER_BIT|GL_STENCIL_BUFFER_BIT)...	
130555	glBindTexture(target=GL_TEXTURE_2D, texture=299)	
130556	glActiveTexture(texture=GL_TEXTURE2)	

State	Value
GL_DEBUG_OUTPUT_SYNCHRO...	GL_FALSE
GL_DEBUG_OUTPUT_SYNCHRO...	GL_FALSE
GL_DEPTH_BITS	<unknown>
GL_DEPTH_CLEAR_VALUE	0.0
GL_DEPTH_FUNC	GL_ALWAYS
GL_DEPTH_RANGE	0.0, 1.0
GL_DEPTH_TEST	GL_TRUE
GL_DEPTH_WRITEMASK	GL_FALSE
GL_DISPATCH_INDIRECT_BUFFE...	0

Message	Count
API call returned an error code.	14
99.98% of the draw calls are using GL_TRIANGLES.	8488
Draw call indices buffer may be too sparse. (Total sparseness > 2.81)	5275

Qualcomm Snapdragon Profiler



```
Drawcall [1]
  Fail: CullFaceFail
Drawcall [2]
  C0: [35,41,82,255]
Drawcall [10]
  C0: [227,228,230,255]
Drawcall [18]
  C0: [185,126,101,28]
Drawcall [21]
  C0: [185,126,101,28]
```

```
ID  Name           Parameters                                                                                      Clocks
15  glDrawElements ( mode = GL_TRIANGLE_STRIP, count = 2416, type = GL_UNSIGNED_SHORT, indices = 0x0 )             17282
16  glDrawElements ( mode = GL_TRIANGLE_STRIP, count = 2416, type = GL_UNSIGNED_SHORT, indices = 0x0 )             157054
17  glDrawElements ( mode = GL_TRIANGLE_STRIP, count = 2416, type = GL_UNSIGNED_SHORT, indices = 0x0 )             12277
18  glDrawElements ( mode = GL_TRIANGLES, count = 27678, type = GL_UNSIGNED_SHORT, indices = 0x0 )                 254423
19  glDrawElements ( mode = GL_TRIANGLES, count = 9528, type = GL_UNSIGNED_SHORT, indices = 0x0 )                  130169
20  glClear        ( mask = COLOR|DEPTH )                                                                          130169
21  glDrawElements ( mode = GL_TRIANGLES, count = 6, type = GL_UNSIGNED_SHORT, indices = 0x29774664 )              245728
22  glDrawArrays   ( mode = GL_TRIANGLES, first = 0, count = 6 )                                                    25743
```

RenderDoc

RenderDoc

- Widely used on desktop
  - DirectX, OpenGL & Vulkan
- Android support in progress
  - Vulkan and OpenGL ES
  - Alpha support in latest nightly build

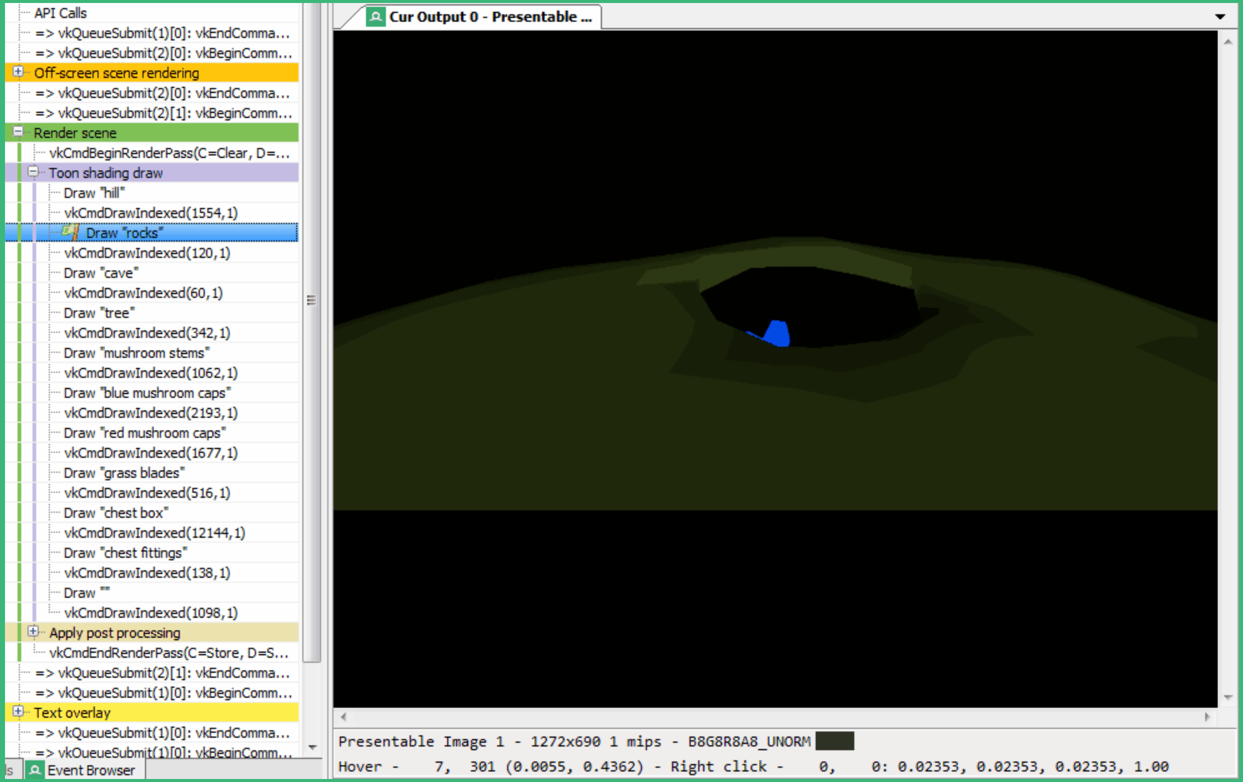
RenderDoc, Vulkan & Android: Components

- Vulkan layer
  - Must be packaged in your game APK
  - Some game permission required, e.g. INTERNET
- Device-side server
  - Server APK must be installed
  - Responsible for communicating with the GUI
  - Also responsible for frame playback

RenderDoc, Vulkan & Android: Capture & replay

- Single frame capture
- Server replays the frame
  - Retrieves GPU output, e.g. rendered images when draw call scrubbing
- Must be replayed on the same device as capture

RenderDoc GUI



```
API Calls
=> vkQueueSubmit(1)[0]: vkEndComma...
=> vkQueueSubmit(2)[0]: vkBeginComm...
Off-screen scene rendering
=> vkQueueSubmit(2)[0]: vkEndComma...
=> vkQueueSubmit(2)[1]: vkBeginComm...
Render scene
  vkCmdBeginRenderPass(C=Clear, D=...
  Toon shading draw
    Draw "hill"
    vkCmdDrawIndexed(1554,1)
    Draw "rocks"
    vkCmdDrawIndexed(120,1)
    Draw "cave"
    vkCmdDrawIndexed(60,1)
    Draw "tree"
    vkCmdDrawIndexed(342,1)
    Draw "mushroom stems"
    vkCmdDrawIndexed(1062,1)
    Draw "blue mushroom caps"
    vkCmdDrawIndexed(2193,1)
    Draw "red mushroom caps"
    vkCmdDrawIndexed(1677,1)
    Draw "grass blades"
    vkCmdDrawIndexed(516,1)
    Draw "chest box"
    vkCmdDrawIndexed(12144,1)
    Draw "chest fittings"
    vkCmdDrawIndexed(138,1)
    Draw ""
    vkCmdDrawIndexed(1098,1)
  Apply post processing
  vkCmdEndRenderPass(C=Store, D=S...
=> vkQueueSubmit(2)[1]: vkEndComma...
=> vkQueueSubmit(1)[0]: vkBeginComm...
Text overlay
=> vkQueueSubmit(1)[0]: vkEndComma...
=> vkQueueSubmit(1)[0]: vkBeginComm...
```

```
Presentable Image 1 - 1272x690 1 mips - B8G8R8A8_UNORM
Hover -    7,  301 (0.0055, 0.4362) - Right click -    0,    0: 0.02353, 0.02353, 0.02353, 1.00
```

Summary

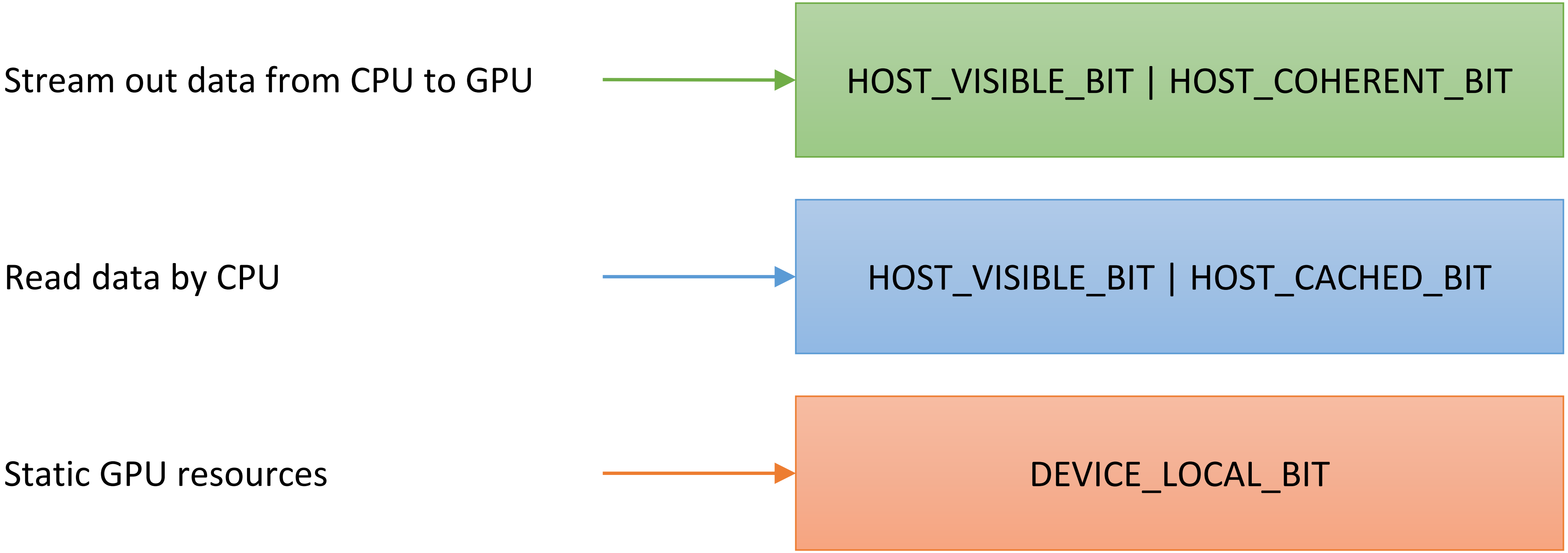
- Wide variety of Google, OEM, IHV tools available
- Cross-platform Vulkan tools, such as GAPID & RenderDoc, are maturing rapidly

Vulkan Optimization Case Studies

Jungwoo Kim

Memory Management

You can create the memory according to your purpose!



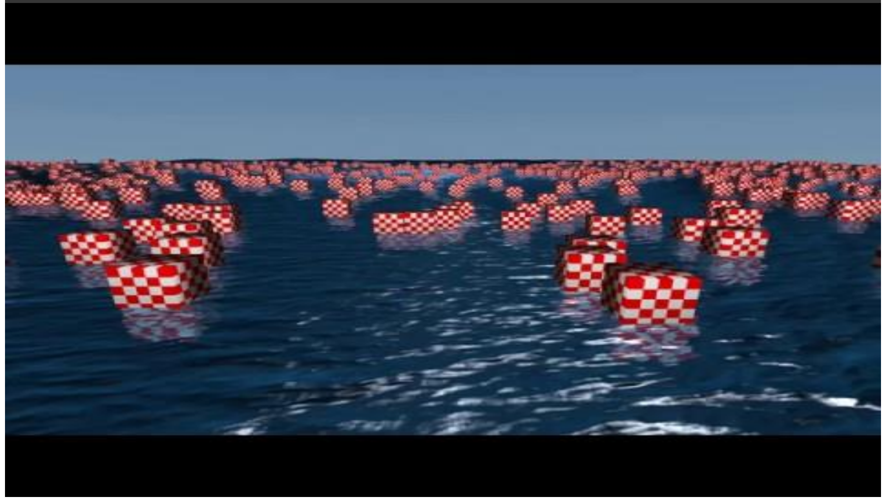
Stream out data from CPU to GPU → HOST_VISIBLE_BIT | HOST_COHERENT_BIT

Read data by CPU → HOST_VISIBLE_BIT | HOST_CACHED_BIT

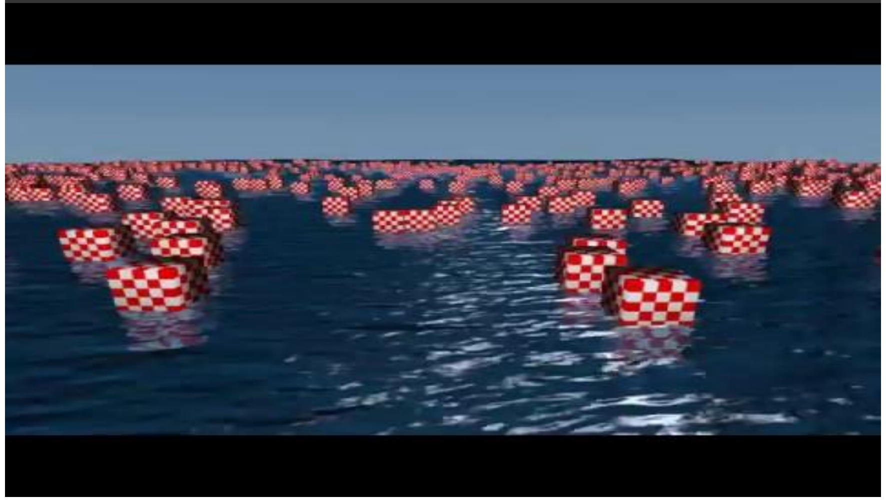
Static GPU resources → DEVICE_LOCAL_BIT

Uniform Buffer

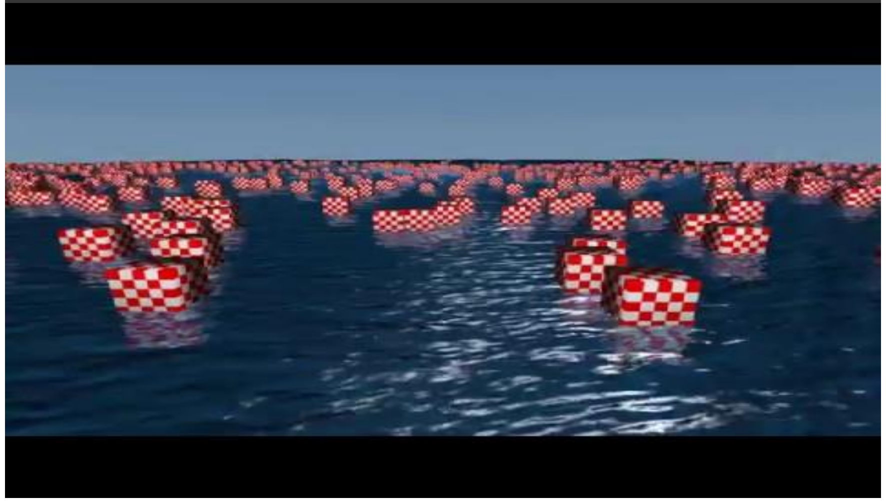
Brute Force 1 FPS



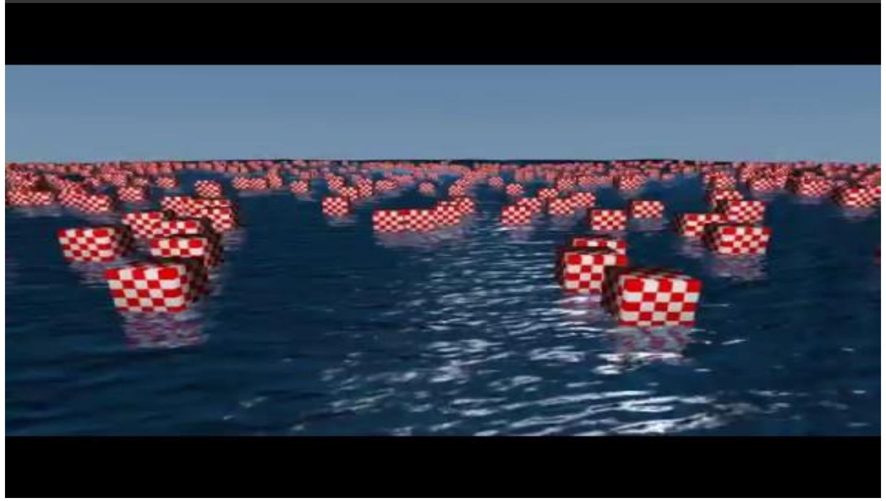
Reusing Memory 37 FPS



Dynamic Offsets 40 FPS



Ideal Condition 43 FPS



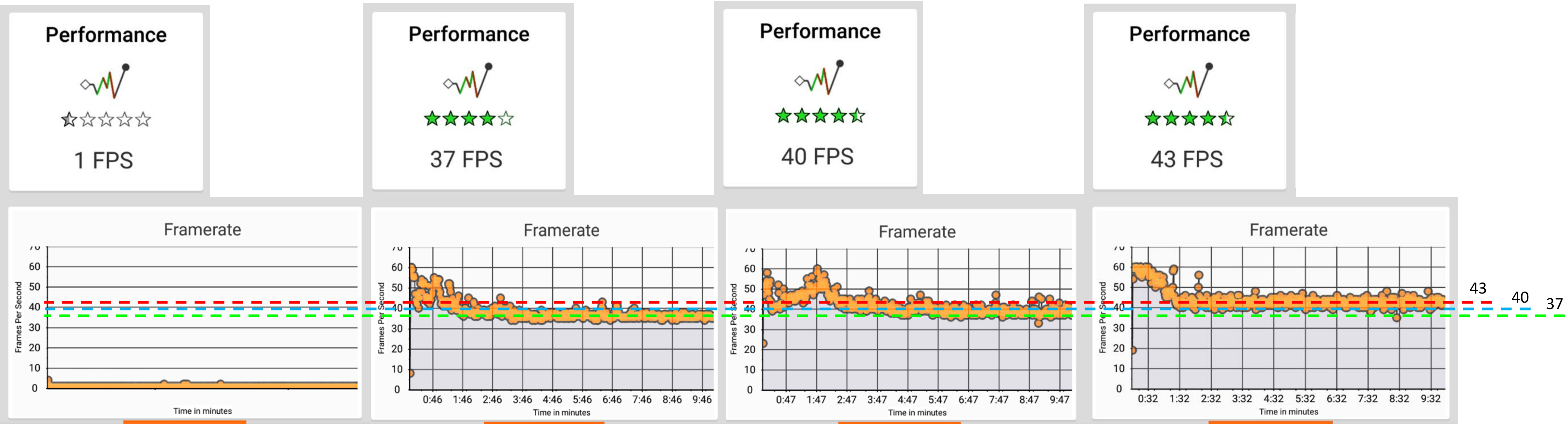
Uniform Buffer

1st Brute Force

2nd Memory Manager

3rd Dynamic Offsets

4th Ideal condition



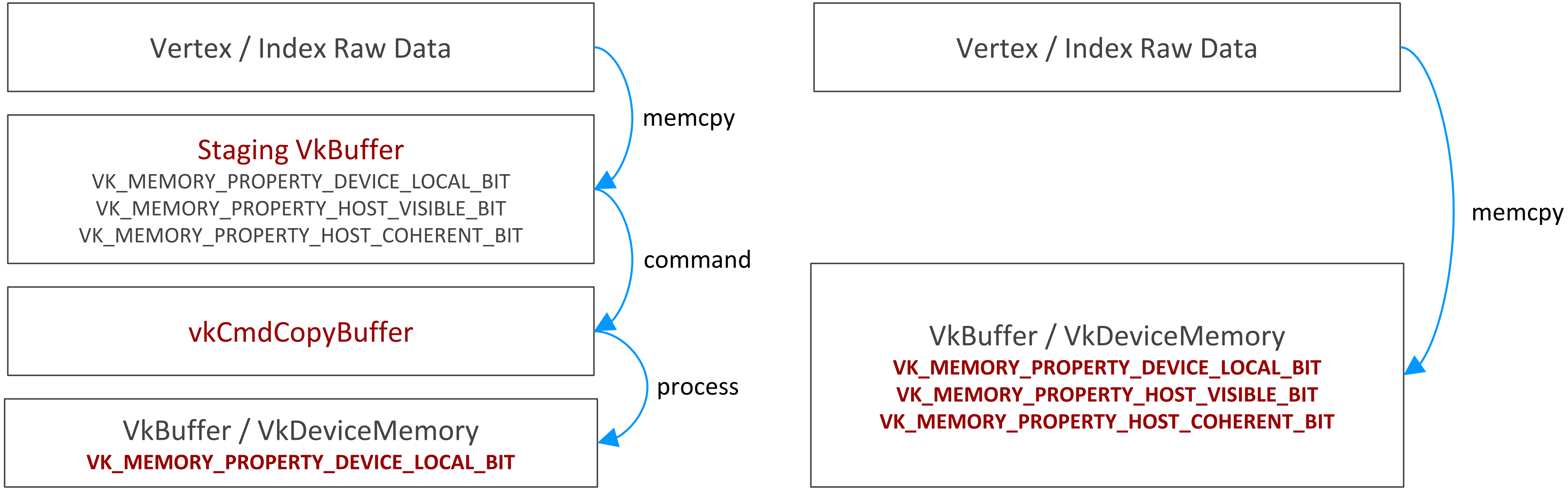
Remember : Structural selection depends on your renderer interface.
Please use these result for reference only.

1st Brute Force : Create Buffer and Allocate Memory in every draw call.
2nd Memory Manager : Use memory manager for reusing VkBuffer and VkDeviceMemory.
3rd Dynamic Offsets : Also use memory manager but can skip vkUpdateDescriptorSets API with dynamic offsets feature.
4th Ideal condition : If everything is in a predictable situation. There is no overhead for caching resources.

Vertex / Index Buffer

QC, ARM 의견 받고 써야됨.

- In mobile memory, we don’t need to use staging buffer for Vertex/Index buffer.
- For dynamic objects, performance can be decreased with that logic.



Vertex / Index Raw Data

Staging VkBuffer
VK_MEMORY_PROPERTY_DEVICE_LOCAL_BIT
VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT
VK_MEMORY_PROPERTY_HOST_COHERENT_BIT

vkCmdCopyBuffer

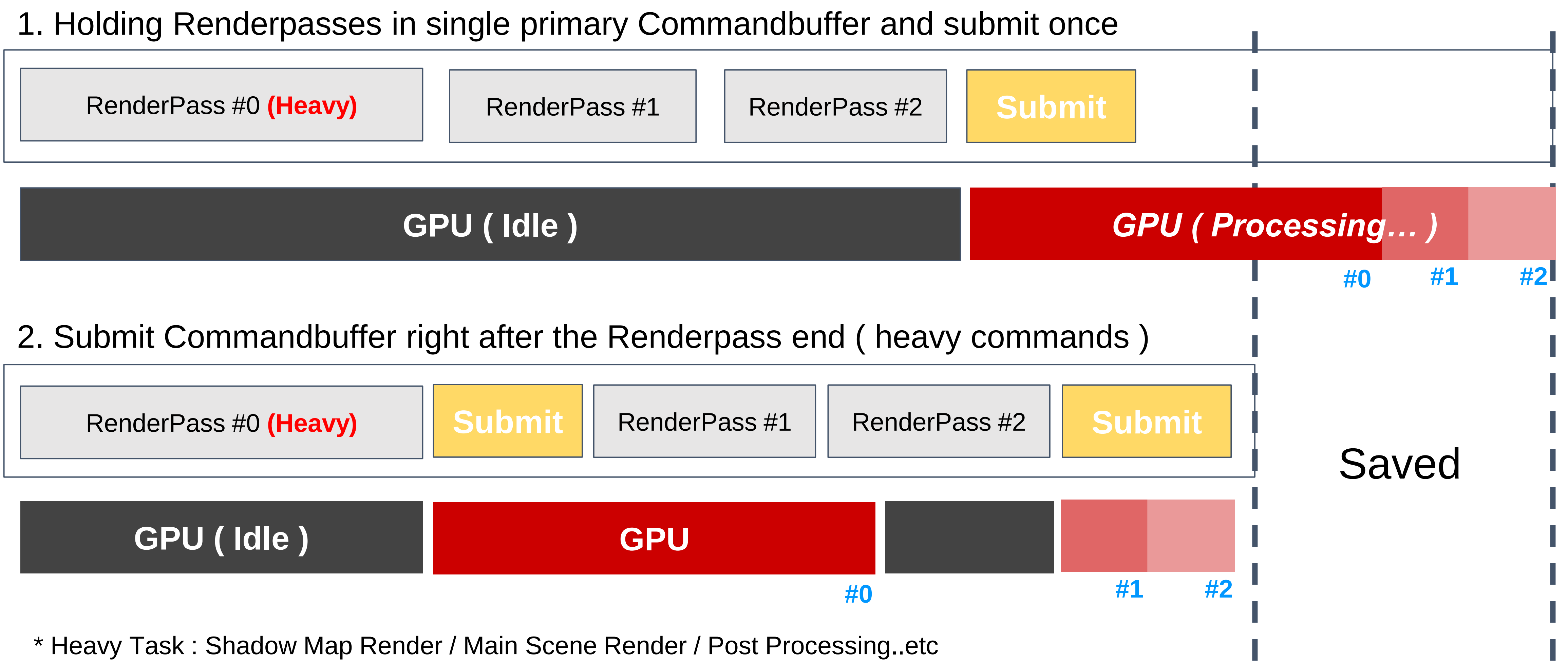
VkBuffer / VkDeviceMemory
VK_MEMORY_PROPERTY_DEVICE_LOCAL_BIT

Vertex / Index Raw Data

VkBuffer / VkDeviceMemory
VK_MEMORY_PROPERTY_DEVICE_LOCAL_BIT
VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT
VK_MEMORY_PROPERTY_HOST_COHERENT_BIT

제가 알기로는 모바일에서 Device Local BIT만 썼을때 얻는 Performance 이득이 없습니다.

Command Buffer (Submit Control)



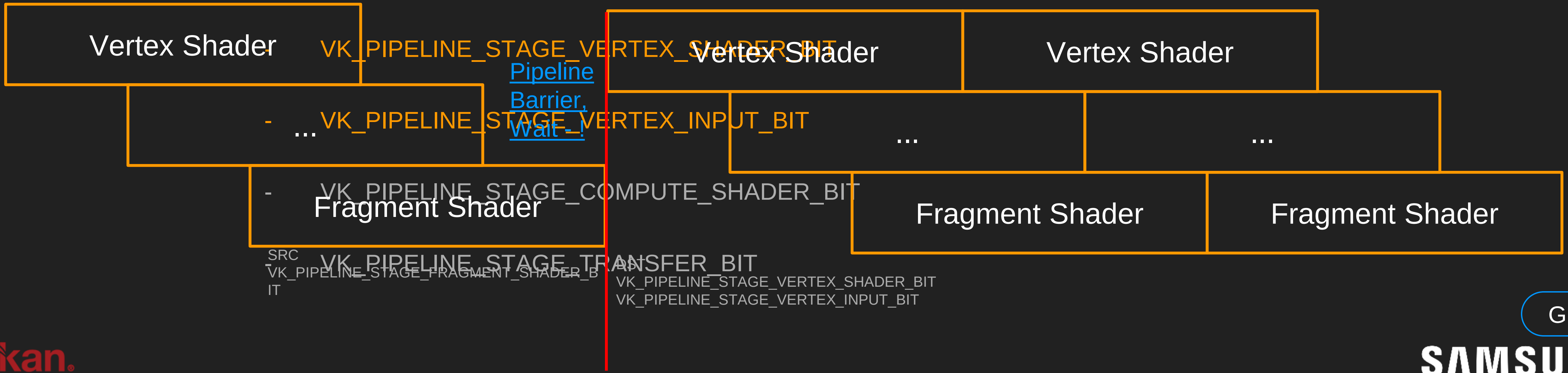
1. Holding Renderpasses in single primary Commandbuffer and submit once

2. Submit Commandbuffer right after the Renderpass end (heavy commands)

* Heavy Task : Shadow Map Render / Main Scene Render / Post Processing..etc

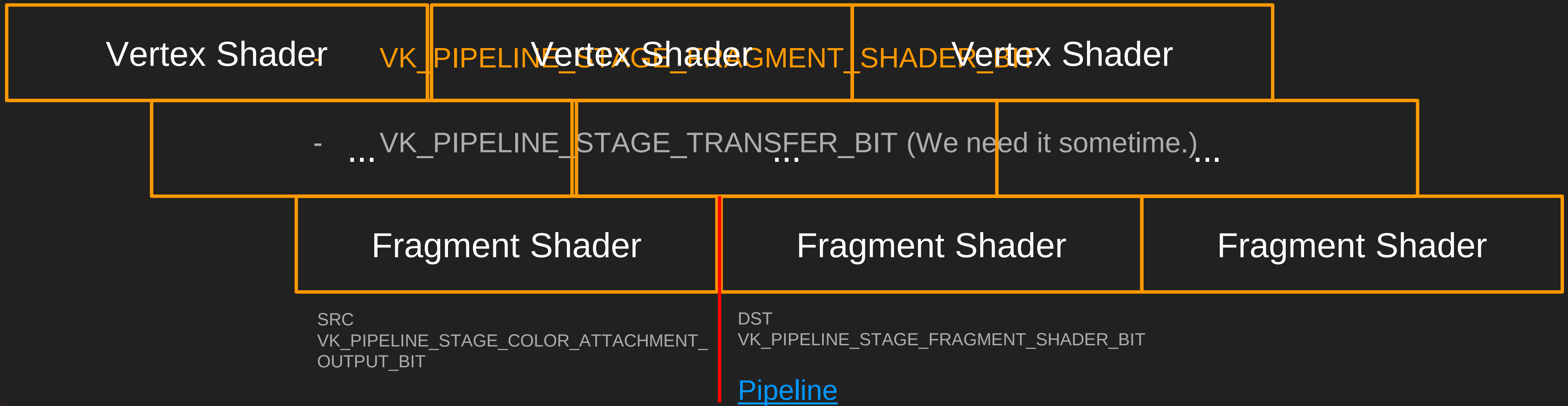
Optimization List - Pipeline Barrier

- Change image layout to readable
  - Wrong stage mask
    - SRC
      - VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT
    - DST
      - VK_PIPELINE_STAGE_VERTEX_SHADER_BIT
      - VK_PIPELINE_STAGE_VERTEX_INPUT_BIT
      - VK_PIPELINE_STAGE_COMPUTE_SHADER_BIT
      - VK_PIPELINE_STAGE_TRANSFER_BIT


Vertex Shader
...
Fragment Shader
Pipeline Barrier, Wait –!
Vertex Shader
Vertex Shader
...
...
Fragment Shader
Fragment Shader
SRC
VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT
DST
VK_PIPELINE_STAGE_VERTEX_SHADER_BIT
VK_PIPELINE_STAGE_VERTEX_INPUT_BIT
GPU

Optimization List - Pipeline Barrier

- Change image layout to readable
  - Correct stage mask
    - SRC
      - VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT
    - DST
      - VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT
      - VK_PIPELINE_STAGE_TRANSFER_BIT (We need it sometime.)



SRC
VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT

DST
VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT

Pipeline Barrier