Frederic Garnier, Samsung Electronics

Bringing Vainglory to Vulkan

f.garnier@samsung.com

@fgarnier_

SAMSUNG

Bringing Vainglory to Vulkan

- Introduction
- Tips & lessons learned
- Performance optimization tips

Bringing Vainglory to Vulkan: **Introduction**

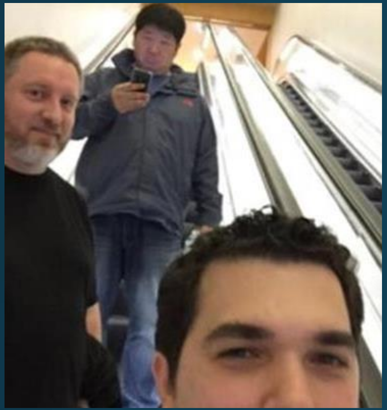
SAMSUNG

Bringing games to Vulkan

- We support game companies & major engine developers
  - We develop cool demos, e.g. ProtoStar
  - Port games to Vulkan such as HIT, Vainglory, NFS etc...

SAMSUNG

Our team



Jonas Gustavsson,
Head of Game Ecosystem
Samsung R&D Institute UK
"Working with Vulkan API developers can finally achieve console-like graphics and high performance on a battery operated device in your pocket."

SAMSUNG

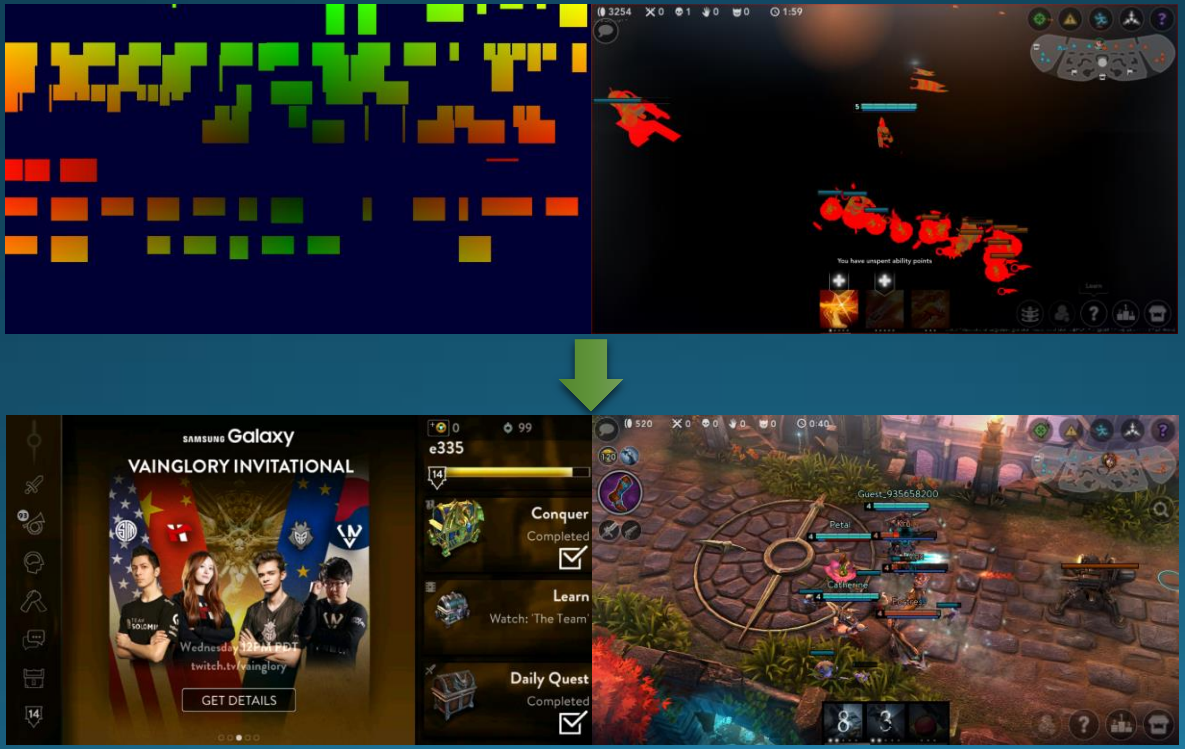
What is Vainglory?

- 3-vs-3 multiplayer online arena battles (MOBA)
- World’s largest mobile eSport

SAMSUNG

Bringing Vainglory to Vulkan

- One of the first Vulkan game we've worked on
- Learned a lot along the way, and there is still more to learn… ☺

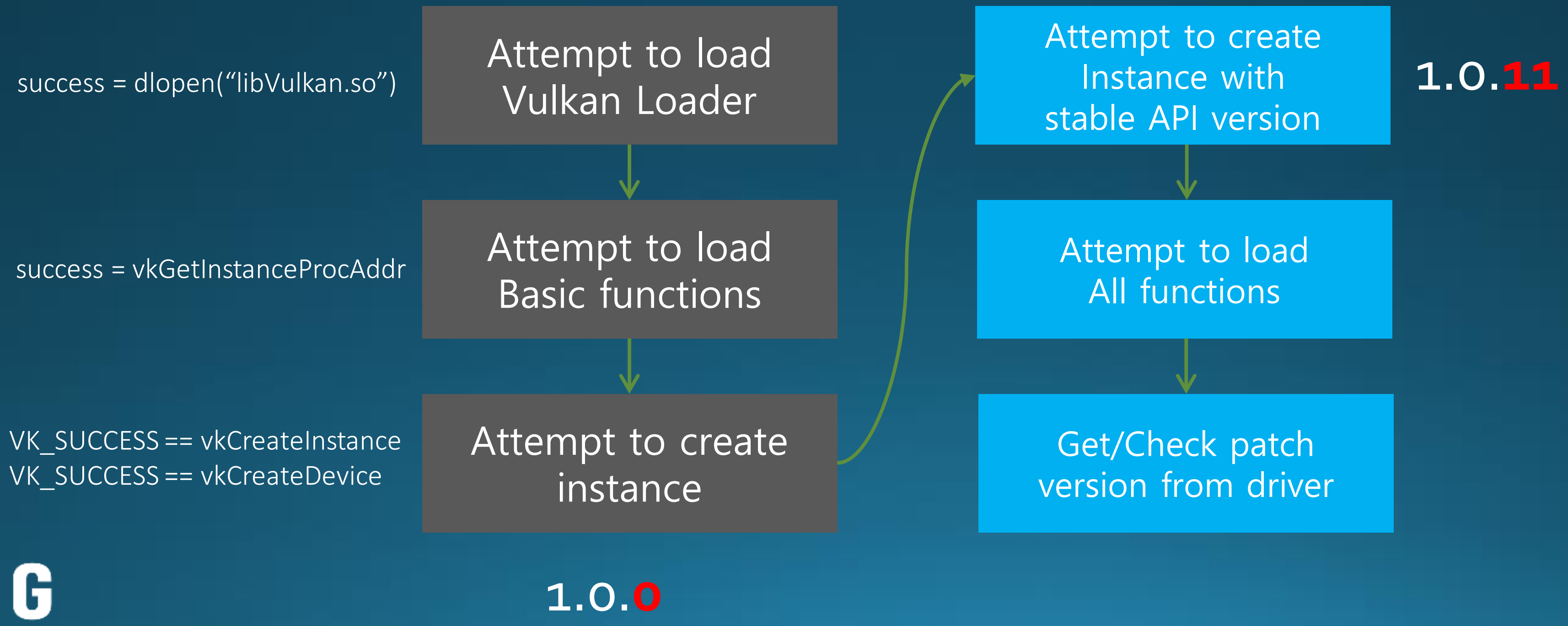


SAMSUNG

Bringing Vainglory to Vulkan: **Tips & lessons learned**

OpenGL ES fallback

- Check for Vulkan available in onCreate() Activity callback
- Fallback to GLES if Vulkan isn't present or version is < 1.0.11

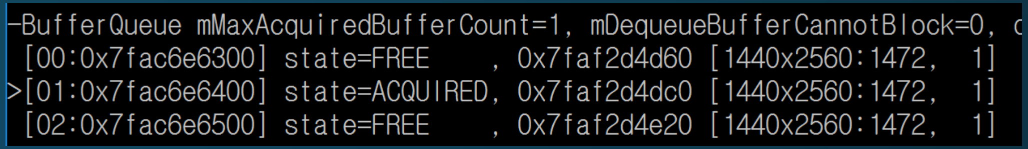


success = dlopen("libVulkan.so")
Attempt to load Vulkan Loader
success = vkGetInstanceProcAddr
Attempt to load Basic functions
VK_SUCCESS == vkCreateInstance
VK_SUCCESS == vkCreateDevice
Attempt to create instance
1.0.0
Attempt to create Instance with stable API version
1.0.11
Attempt to load All functions
Get/Check patch version from driver

SAMSUNG

Swapchain

- Triple Buffer – Google Project Butter (since Android 4.1 Jelly Bean)
  - GLES runs with triple buffering by default
  - adb shell dumpsys SurfaceFlinger →



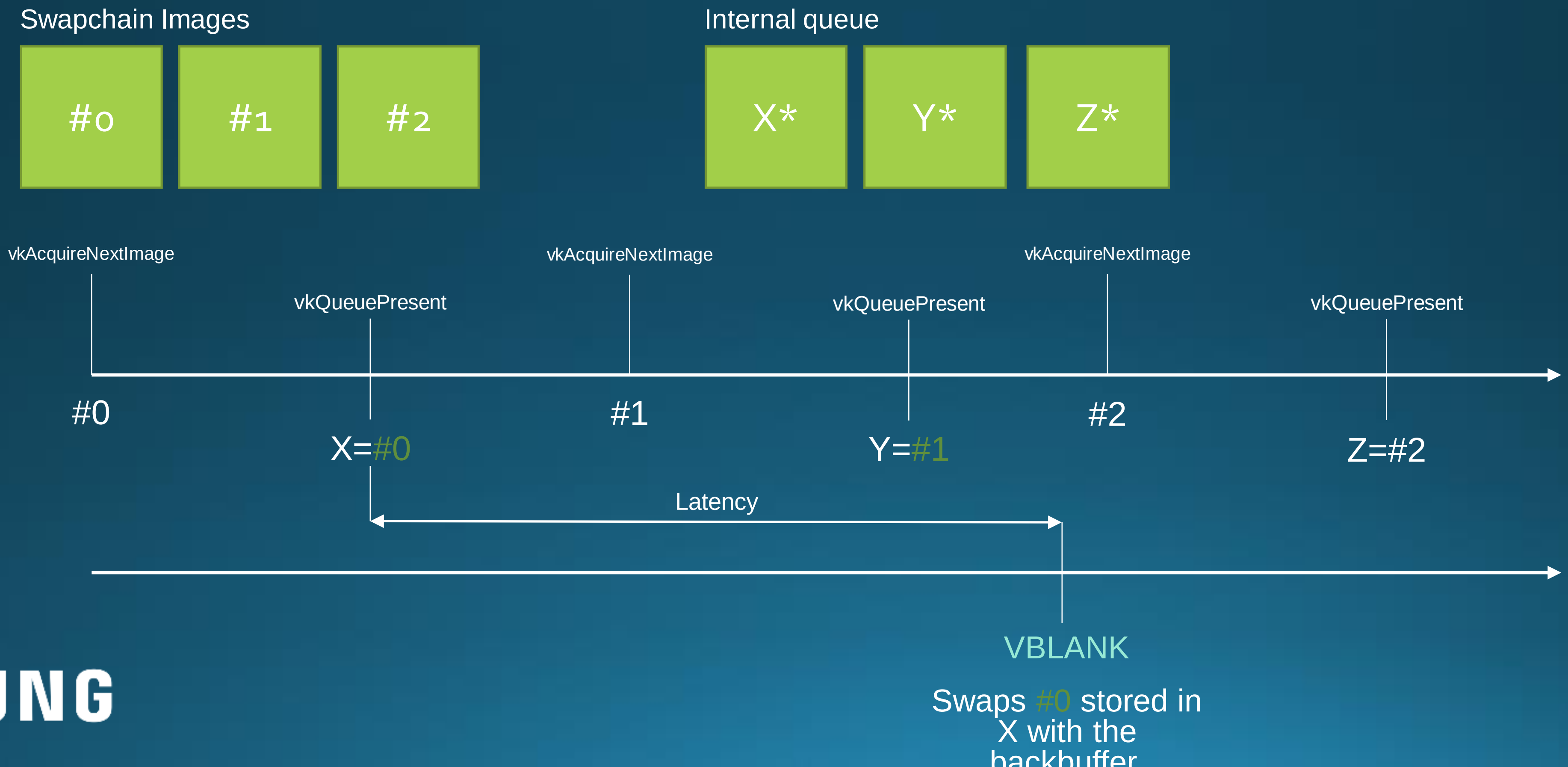
```
-BufferQueue mMaxAcquiredBufferCount=1, mDequeueBufferCannotBlock=0, c
 [00:0x7fac6e6300] state=FREE    , 0x7faf2d4d60 [1440x2560:1472,  1]
>[01:0x7fac6e6400] state=ACQUIRED, 0x7faf2d4dc0 [1440x2560:1472,  1]
 [02:0x7fac6e6500] state=FREE    , 0x7faf2d4e20 [1440x2560:1472,  1]
```

- Android platform requires at least 3 buffers for perf. reasons
  - Can't control the number of back buffers in GLES
- Two present modes available on Android
  - FIFO
  - Mailbox

SAMSUNG

Swapchain: FIFO

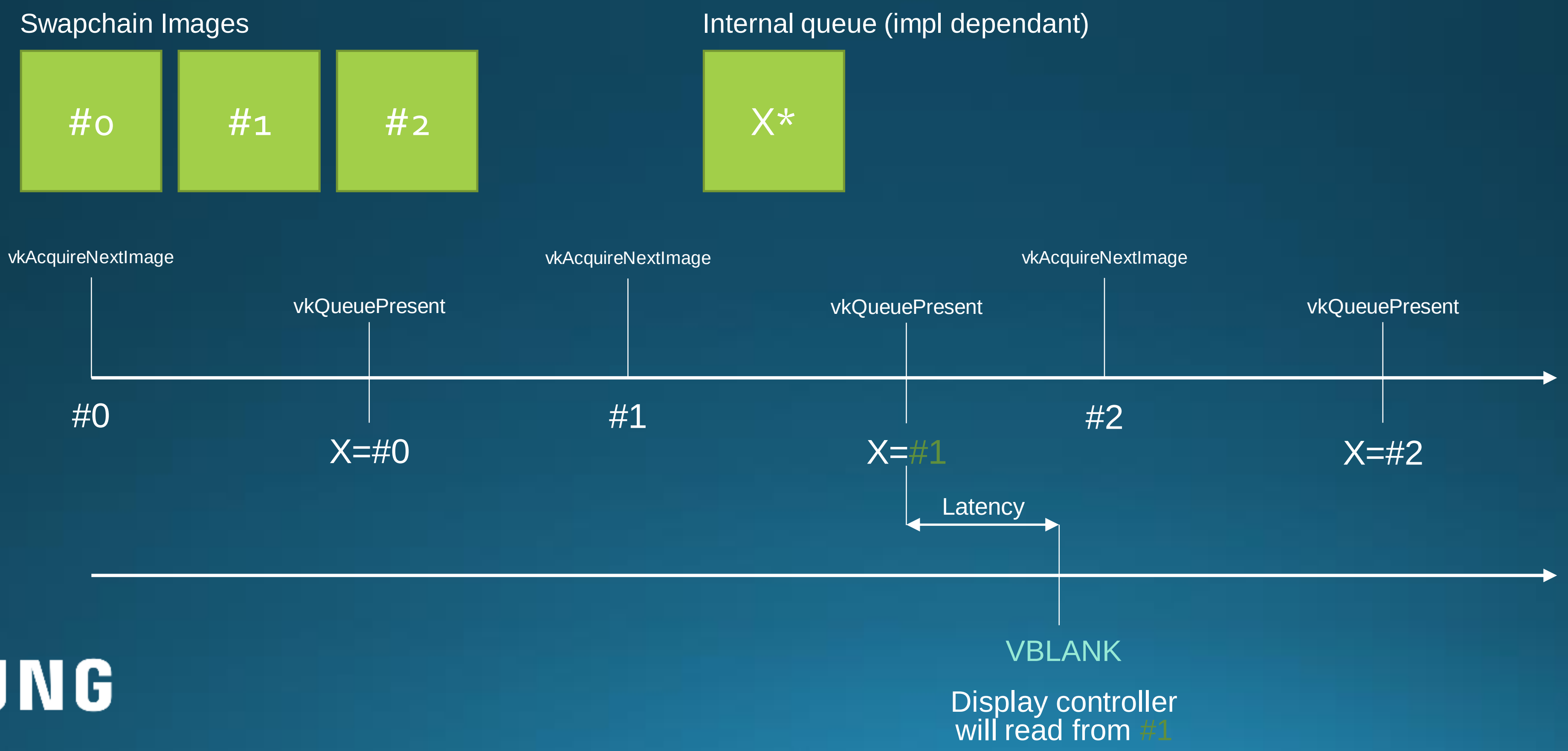
- VK_PRESENT_MODE_FIFO_KHR



SAMSUNG

Swapchain: MAILBOX

- VK_PRESENT_MODE_MAILBOX_KHR



SAMSUNG

Swapchain: MAILBOX vs. FIFO

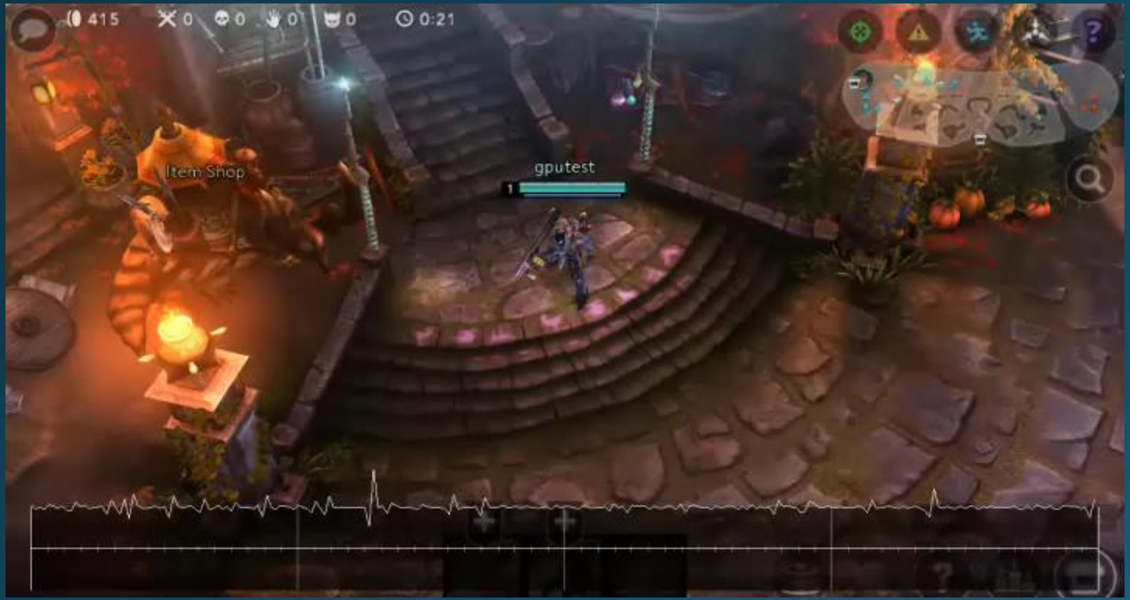
- MAILBOX
  - **Application:** Render to surfaces as fast as possible
  - **Compositor:** Read from the latest surface
  - **Cost:** GPU renders more frames than required ☹

- FIFO
  - **Application:** Blocking wait if surface isn't available
  - **Compositor:** Read from the latest surface
  - **Cost:** GPU only draws what is needed. Compositor reads all surfaces ☺!

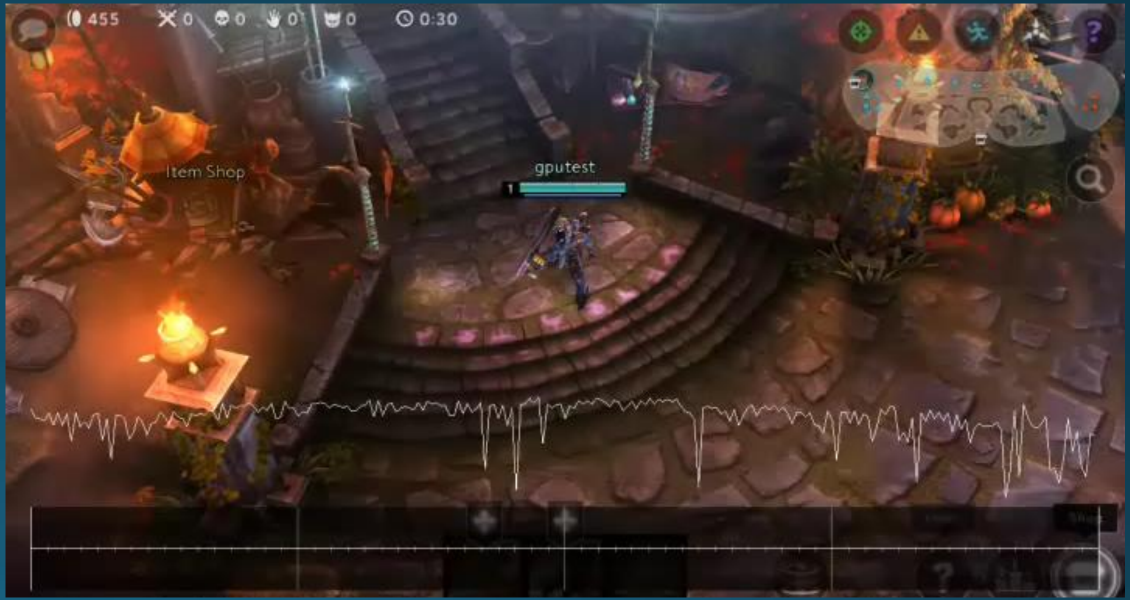
SAMSUNG

Swapchain:
FIFO vs MAILBOX

*VK_PRESENT_MODE_**FIFO**_KHR*



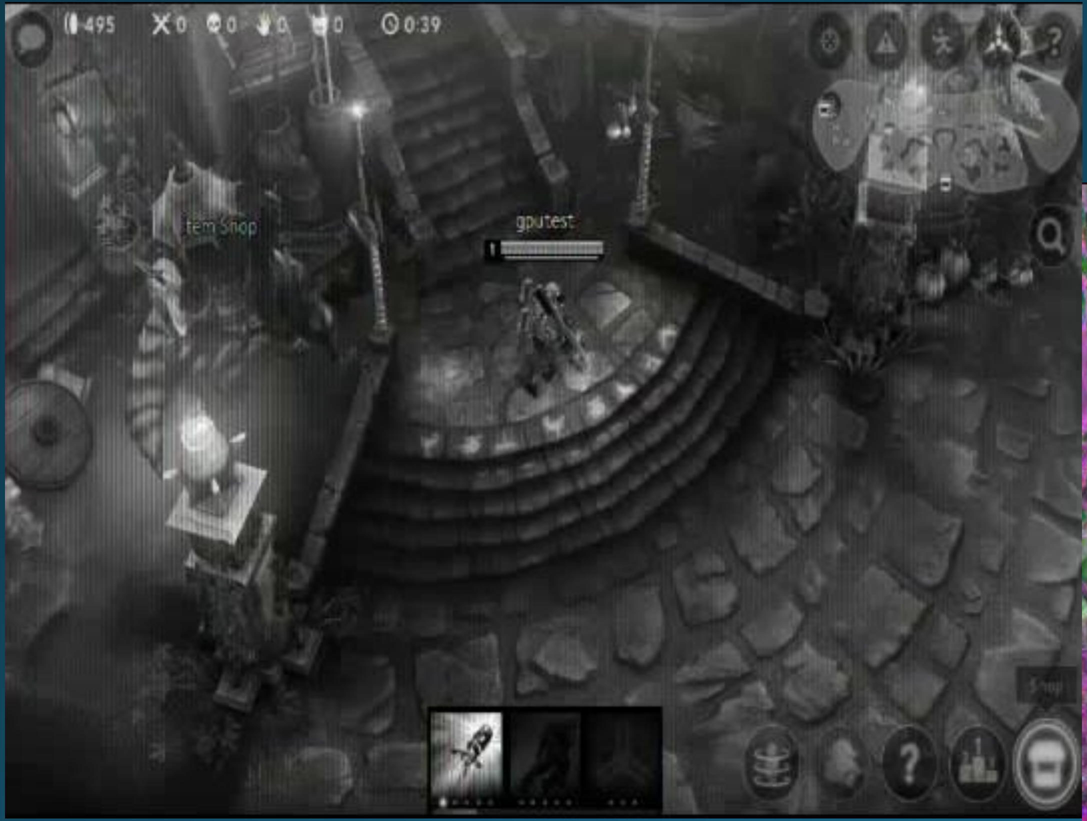
*VK_PRESENT_MODE_**MAILBOX**_KHR*

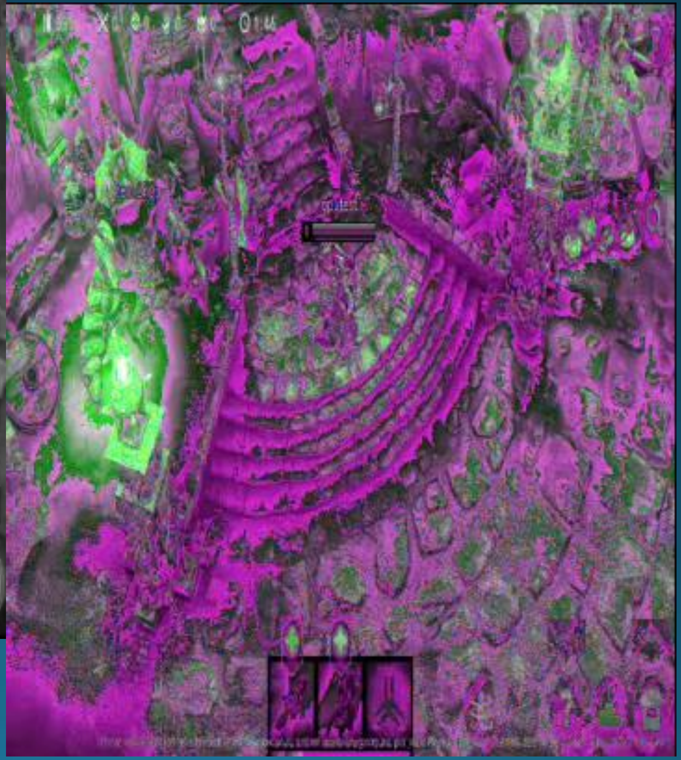


SAMSUNG

Swapchain
Surface format

?

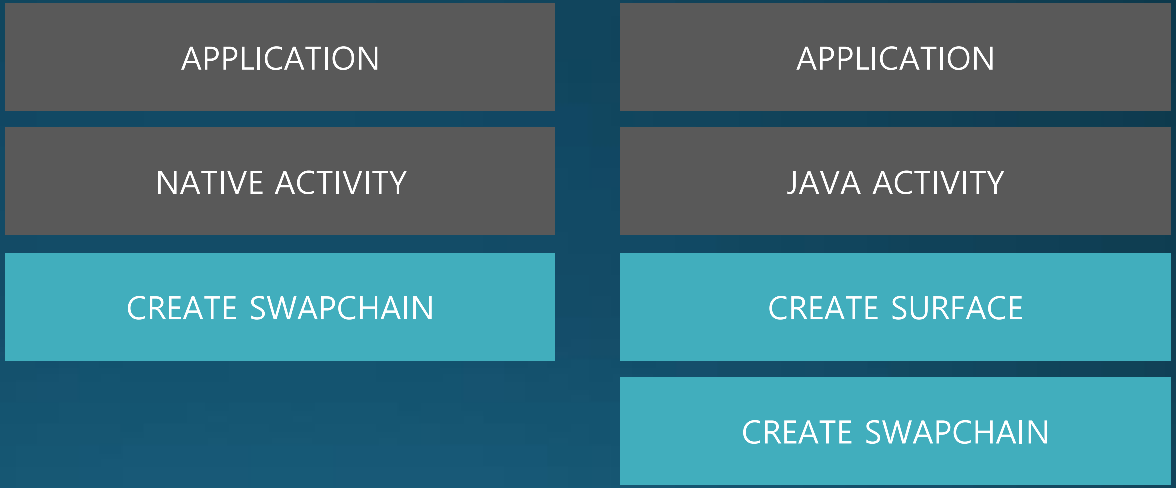




SAMSUNG

Swapchain: Surface format

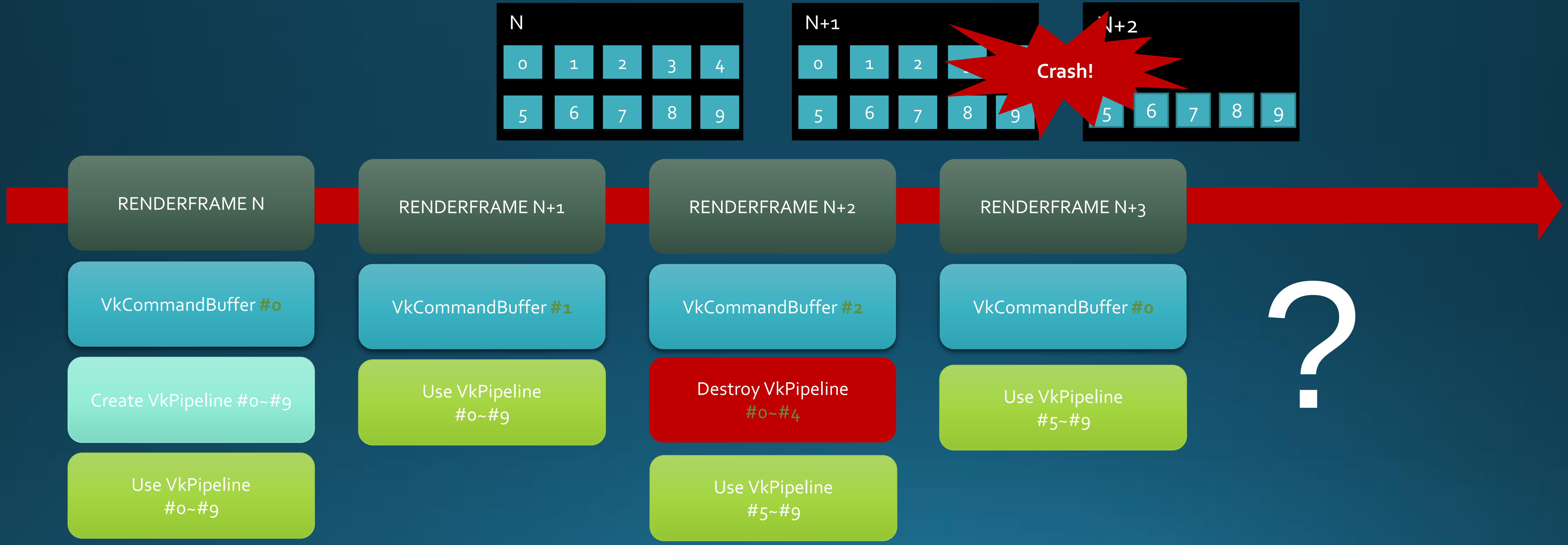
- Java to native
    - All window surfaces must have the same format



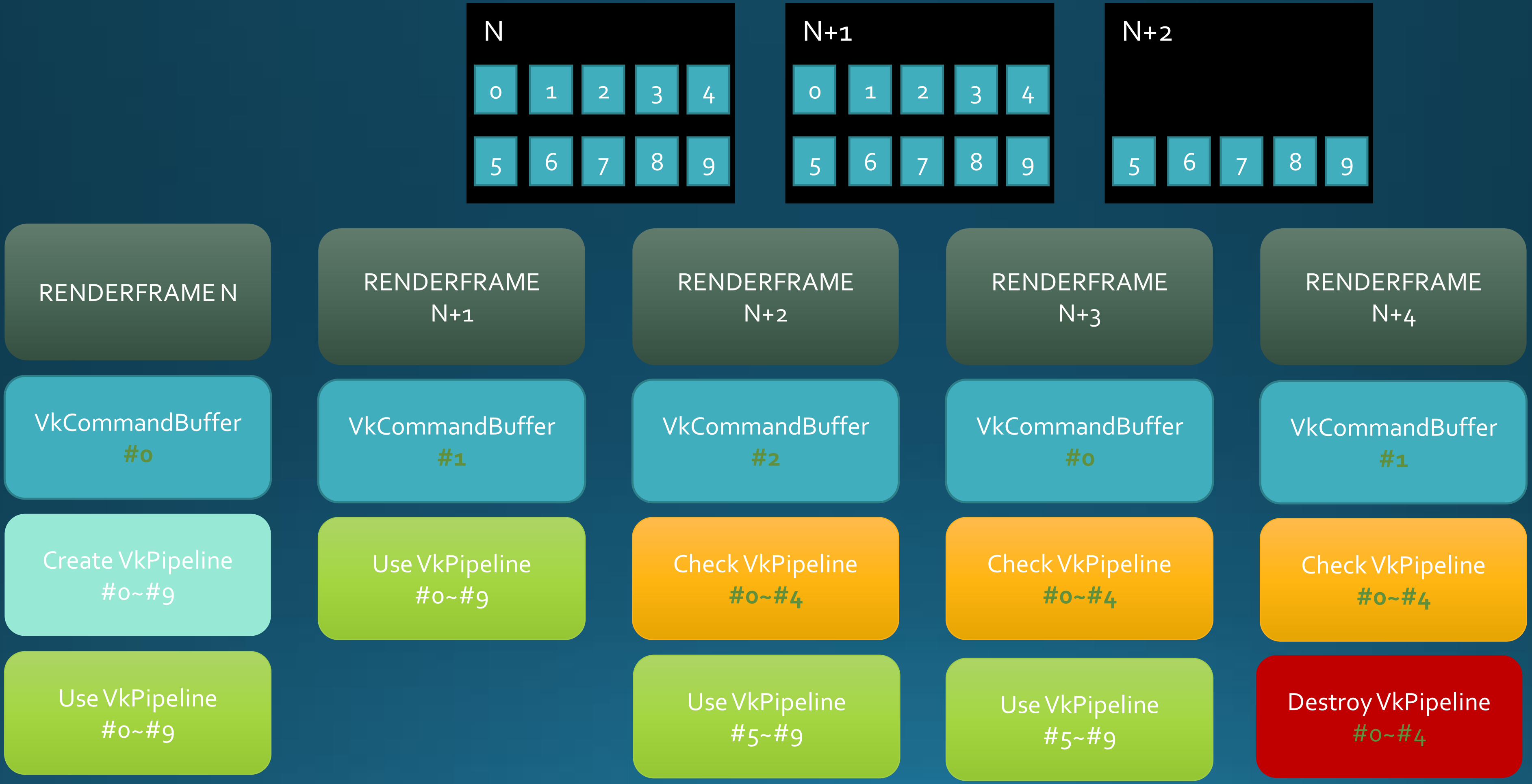
- Example
    - getHolder().setFormat(PixelFormat.RGB_888) must match VK_FORMAT_R8G8B8_UNORM swap chain format

SAMSUNG

Pipeline: Crash on destroy

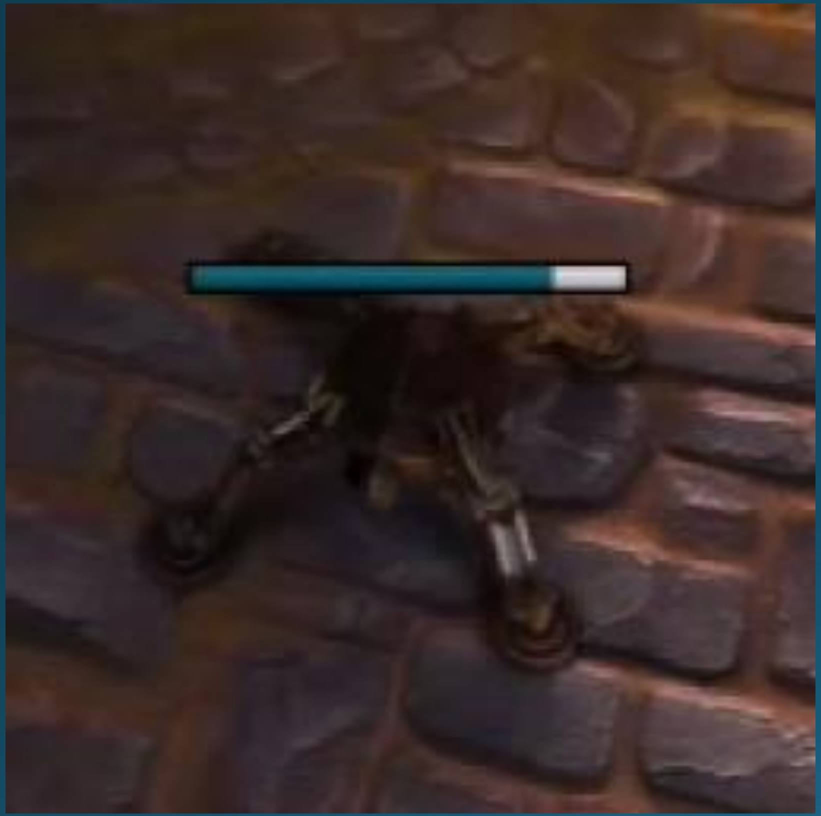
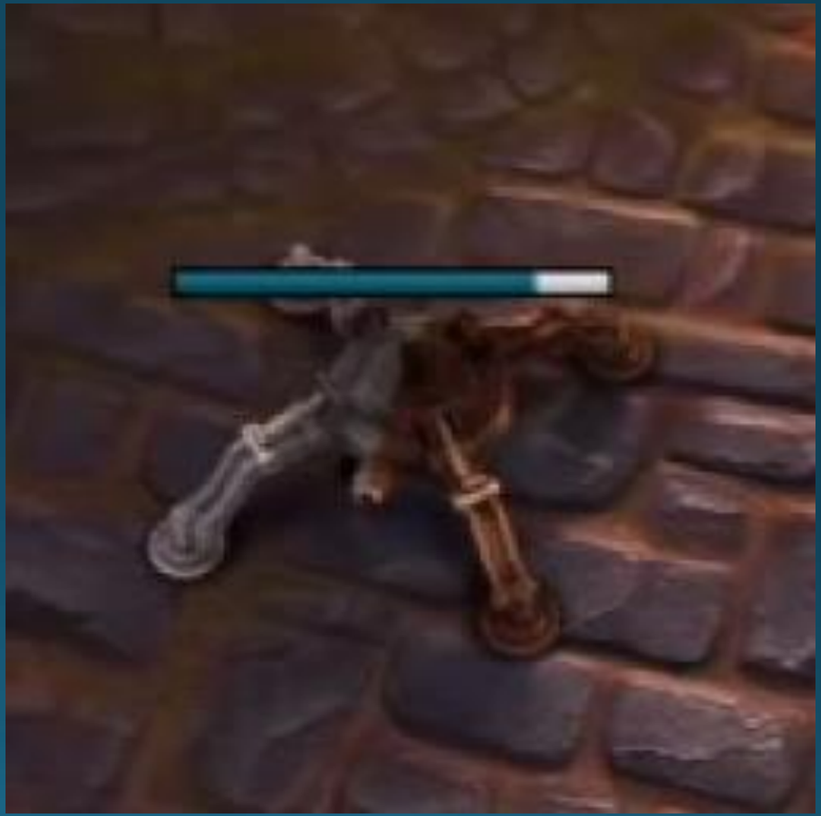


Pipeline: Safe destruction



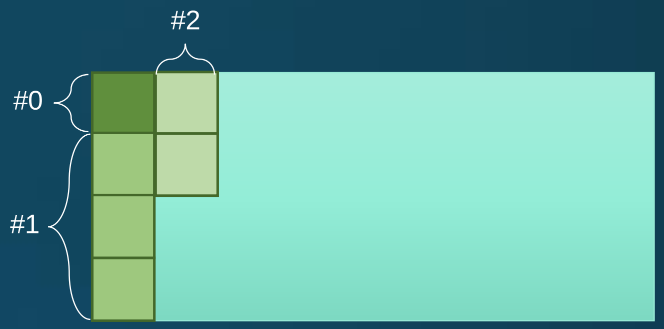
SAMSUNG

Shaders: Memory alignment

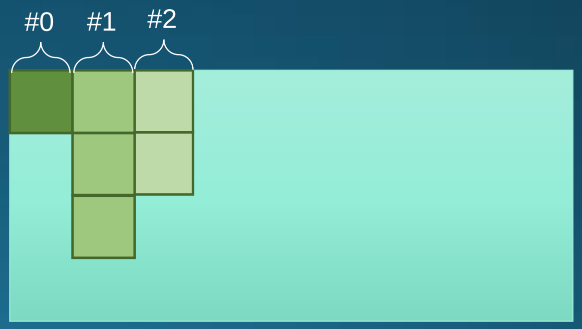
Shaders:
Memory alignment

```
layout(set=0, binding=0) uniform buf1{
  float _unif1; // #0
  vec3  _unif2; // #1
  vec2  _unif3; // #2
}
```



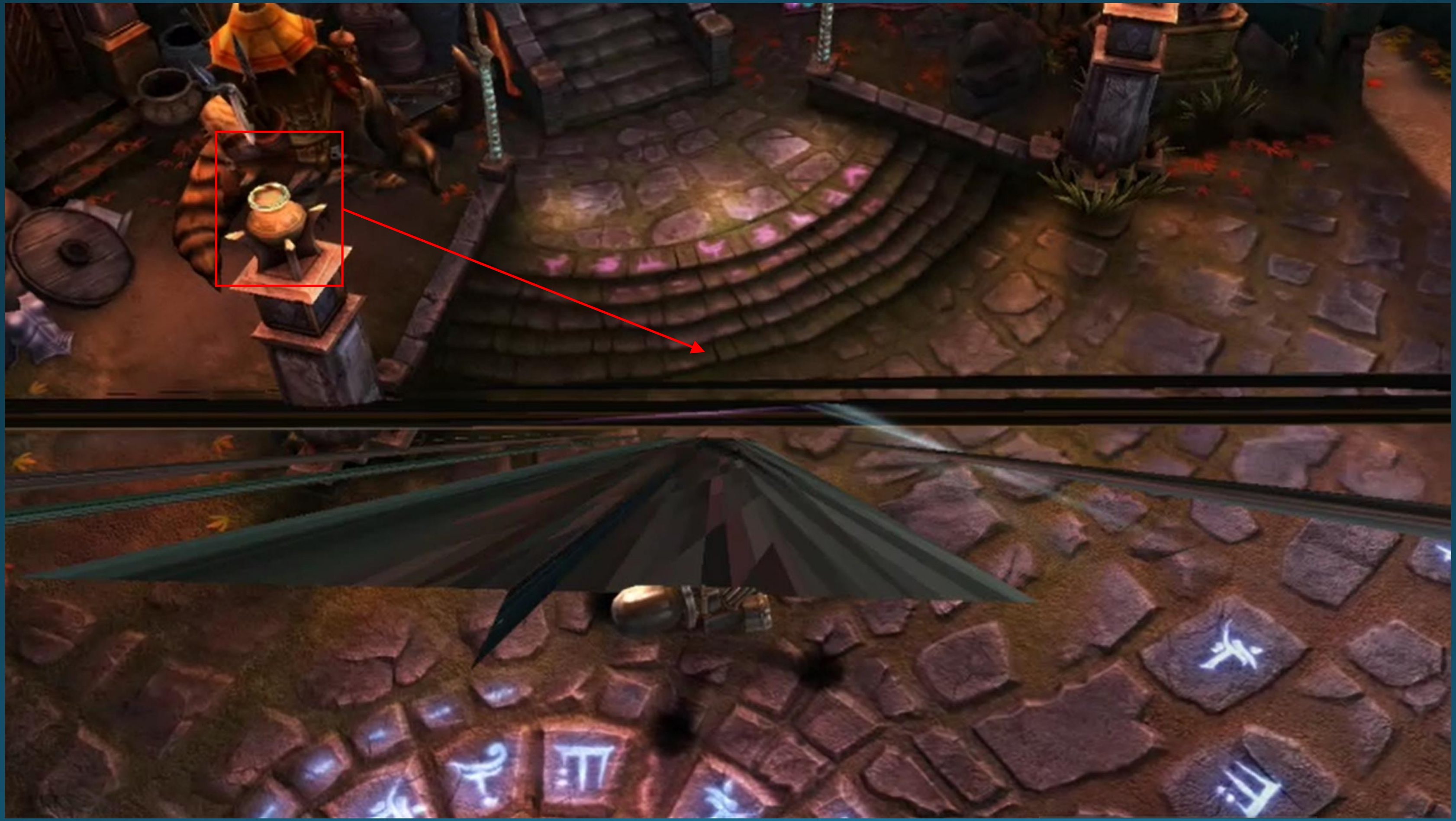
- glslangValidator applies std140 layout by default
- Best to explicitly specify layout in shader code and make sure buffer layout matches!

```
Name 18  "buf1"
MemberName 18(buf1) 0  "_unif1"
MemberName 18(buf1) 1  "_unif2"
MemberName 18(buf1) 2  "_unif3"
Name 20  "ubuf"
MemberDecorate 8(gl_PerVertex) 0 BuiltIn Position
MemberDecorate 8(gl_PerVertex) 1 BuiltIn PointSize
Decorate 8(gl_PerVertex) Block
MemberDecorate 18(buf1) 0 Offset 0
MemberDecorate 18(buf1) 1 Offset 16
MemberDecorate 18(buf1) 2 Offset 32
Decorate 18(buf1) Block
```



SAMSUNG

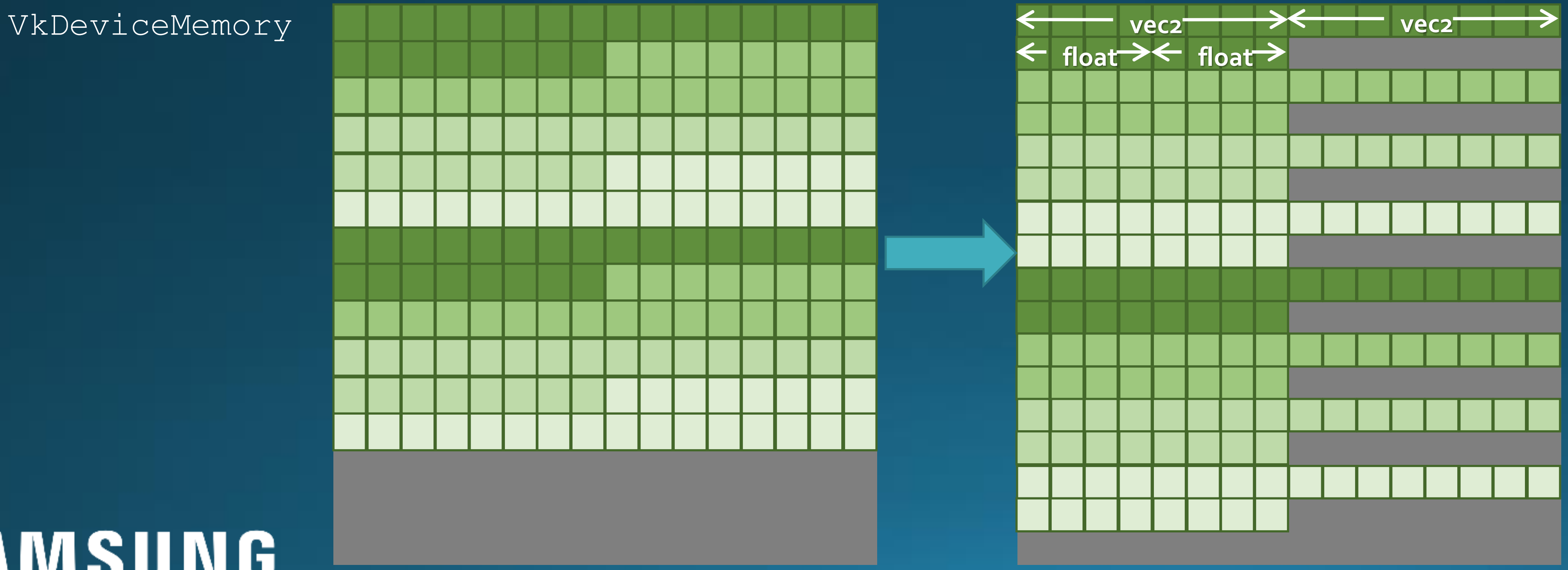
Shaders:
Memory pools



SAMSUNG

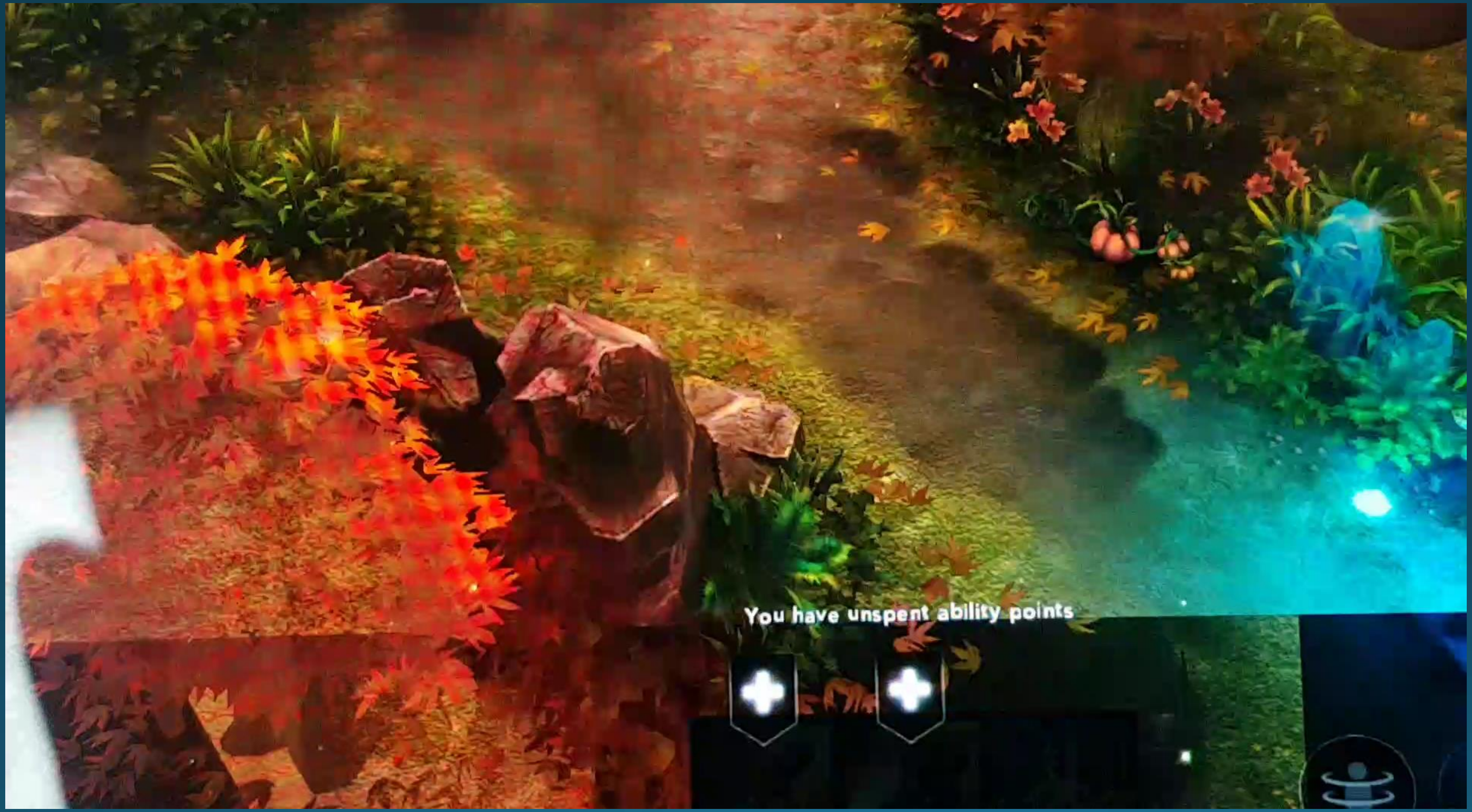
Shaders: Memory pools

- Store uniform data of related draws in one buffer
- Each draw's data must align to VkPhysicalDeviceLimits::*minUniformBufferOffsetAlignment*



SAMSUNG

Uniform buffer: Tiling artefacts

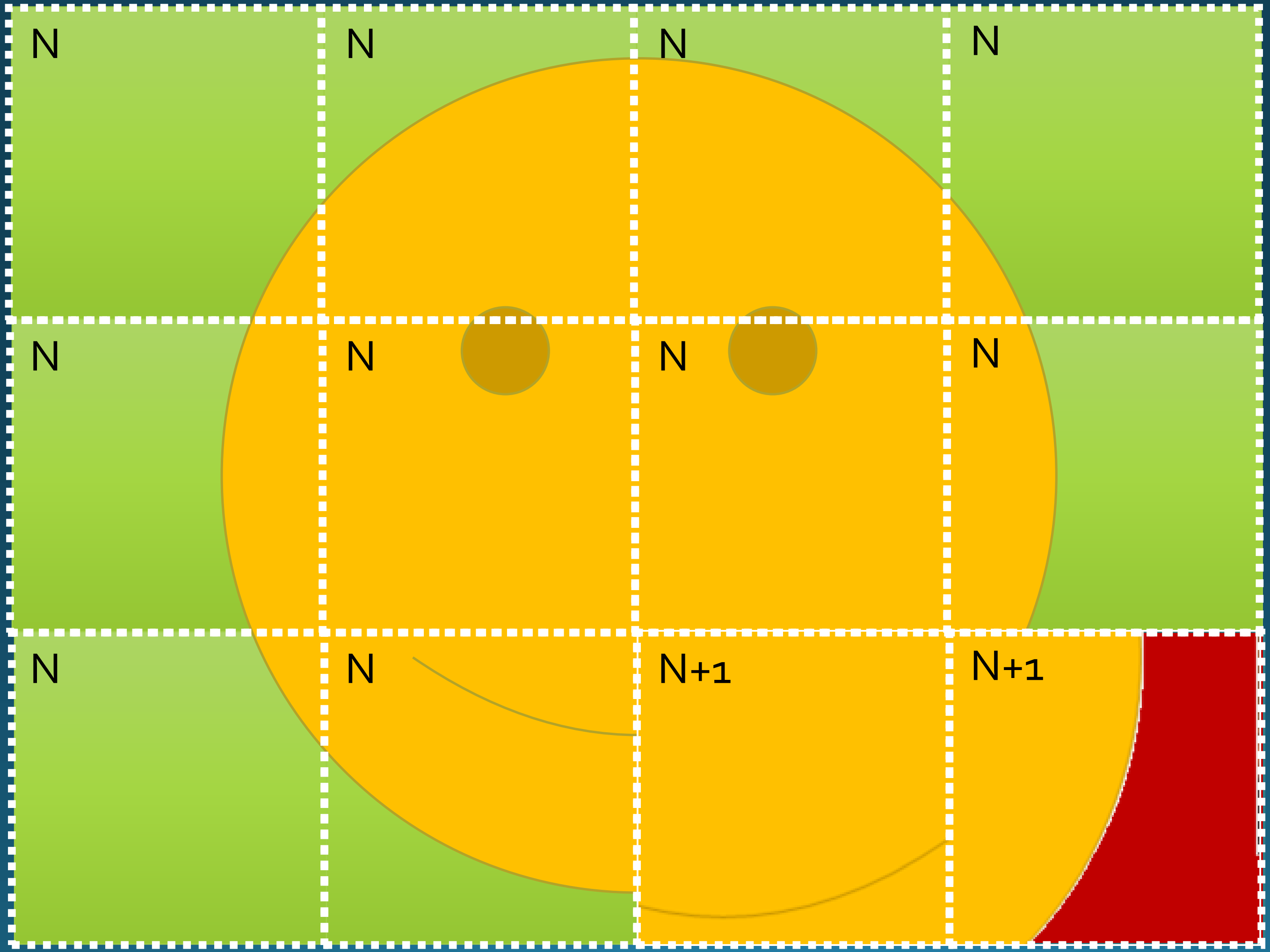


SAMSUNG

Uniform buffer: Tiling artefacts

N = N Frame MVP Matrix

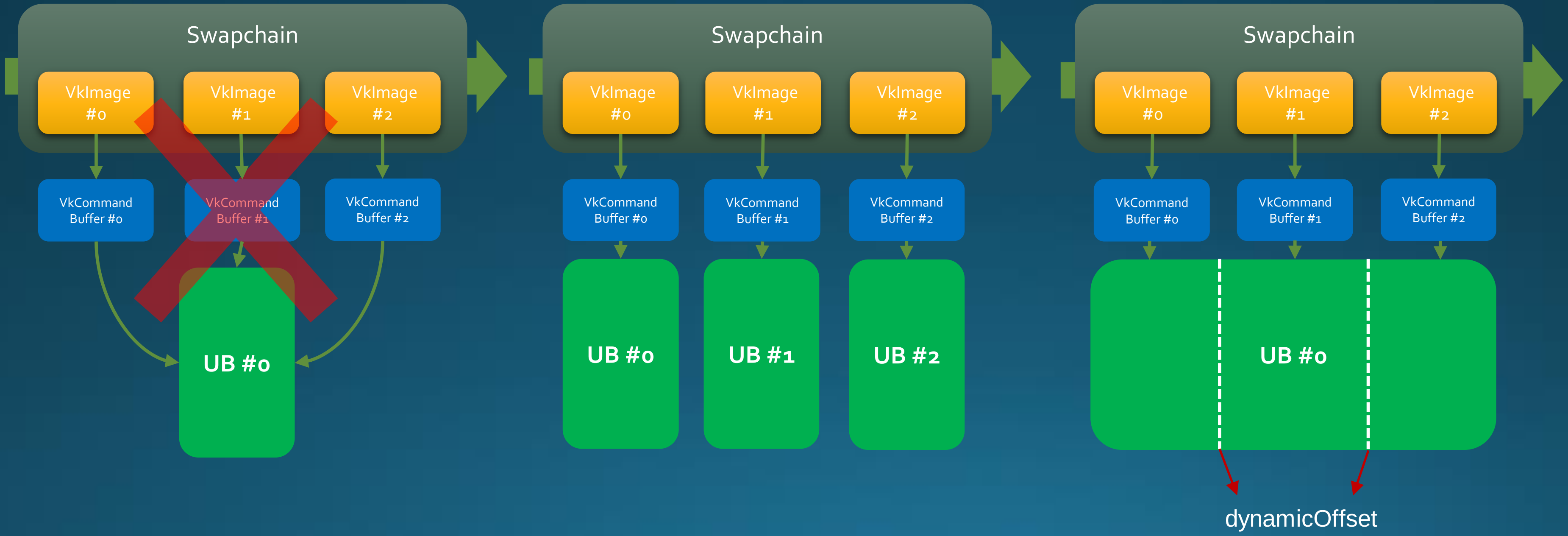
N+1 = N+1 Frame MVP Matrix(Changed)



SAMSUNG

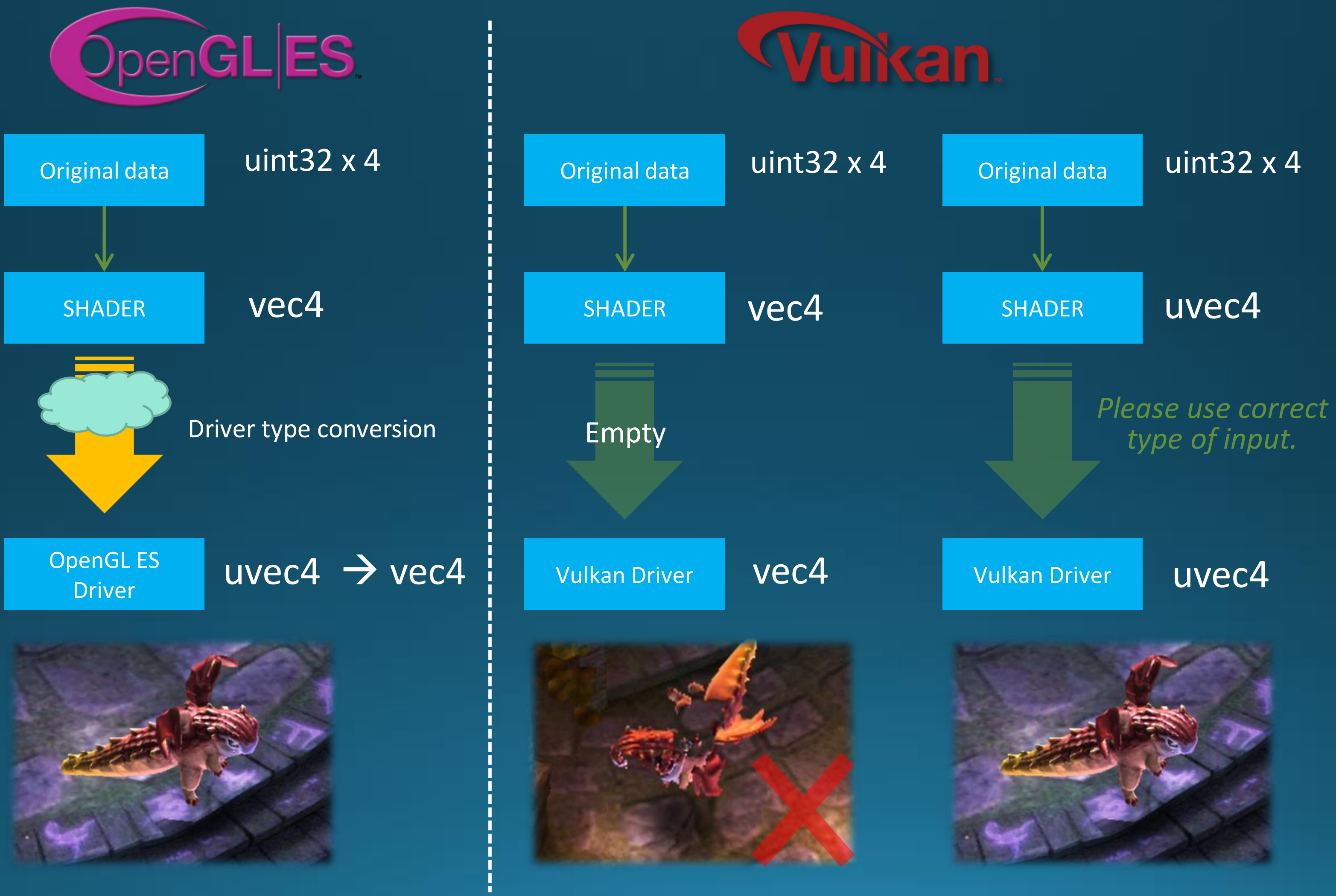
Uniform buffer: Tiling artefacts

- **Option #1:** Uniform buffer per-swapchain index
- **Option #2:** Swapchain index based offset in a single uniform buffer



SAMSUNG

OpenGL ES vs. Vulkan: Type conversions



Bringing Vainglory to Vulkan: **Perf. Optimization tips**

SAMSUNG

Image layouts: Swapchain

- Swapchain layouts
  - Clear with `VK_IMAGE_LAYOUT_GENERAL`
  - Render with `VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL`
  - Present with `VK_IMAGE_LAYOUT_PRESENT_SRC_KHR`

Image layouts: Textures

- VK_TILING_LINEAR vs. VK_TILING_OPTIMAL
  - Check which formats are supported by the physical device
  - If VK_TILING_OPTIMAL is available, use it
    - Requires texture data to be uploaded via a staging buffer
  - If not, use linear
    - Less cache efficient than optimal
    - Map the buffer, modify the data, unmap

```
// Get Image Format Property
VkFormatProperties formatProperty;
vkGetPhysicalDeviceFormatProperties(physicalDevice, imageFormat, &formatProperty);
if (formatProperty.optimalTilingFeatures & VK_FORMAT_FEATURE_SAMPLED_IMAGE_BIT) /**/;
else if (formatProperty.linearTilingFeatures & VK_FORMAT_FEATURE_SAMPLED_IMAGE_BIT) /**/;
```

SAMSUNG

Image layout: Staging buffer

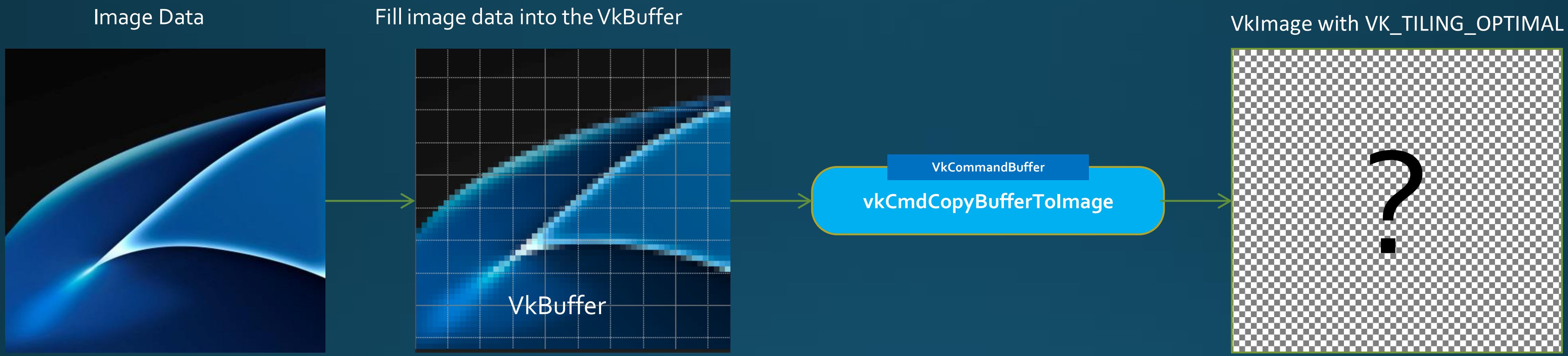


Image Data
Fill image data into the VkBuffer
VkBuffer
VkCommandBuffer
vkCmdCopyBufferToImage
VkImage with VK_TILING_OPTIMAL
?

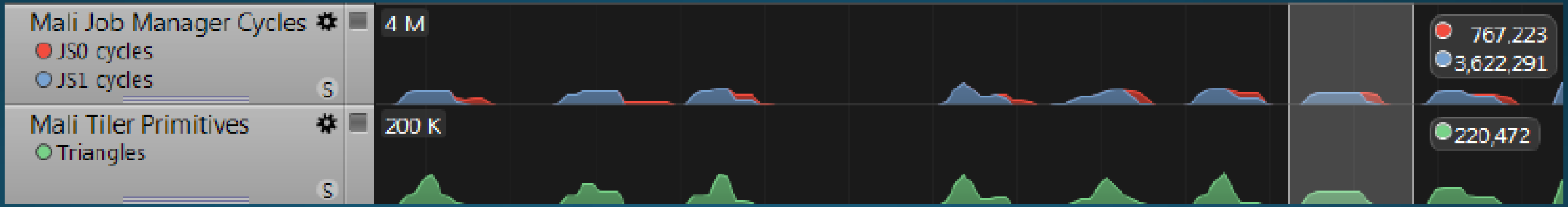
```
VkBuffer& stagingBuffer = getStagingBuffer(imageBufferSize);
VkBufferImageCopy region = getRegionFromImage(image);
fillBuffer(stagingBuffer, pImageData);
vkCmdCopyBufferToImage(commandBuffer, stagingBuffer, image,
VK_IMAGE_LAYOUT_TRANSFER_DST_OPTIMAL, 1, &region);
```

DO NOT use VK_MEMORY_PROPERTY_HOST_VISIBLE_BIT with VK_TILING_OPTIMAL.

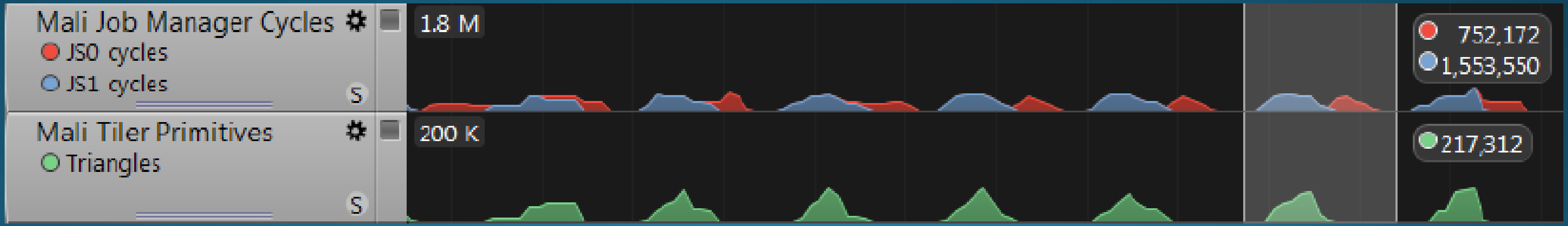
OpenGL ES vs. Vulkan: Geometry sorting

- Geometry sorting (vertex & index buffers)
  - Improves cache read/write efficiency
    - Can affect how work is submitted to the GPU
  - Some OpenGL ES drivers do this automatically

Without Geometry Sorting



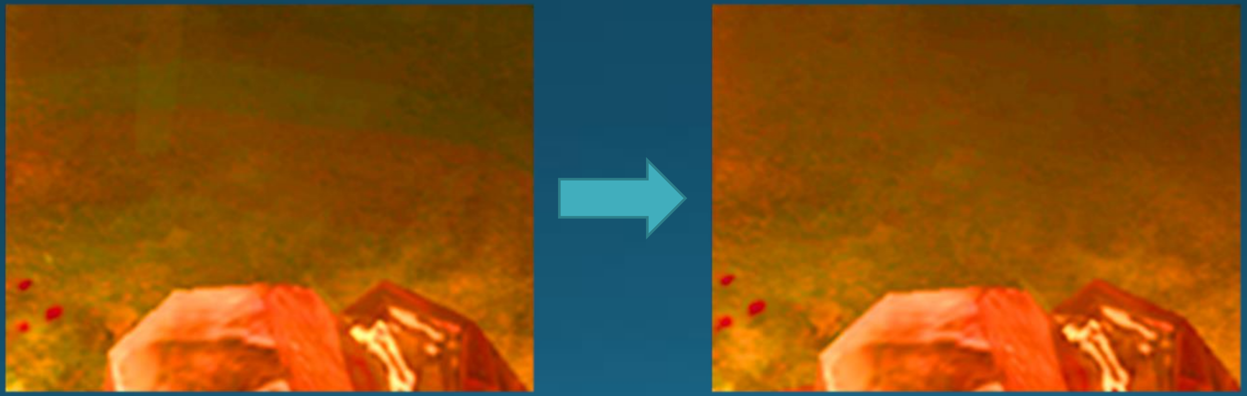
With Geometry Sorting



SAMSUNG

Rendering quality: Blend precision

- Blend precision
  - May be affected by surface format (e.g. RGB565 vs. RGB888)
  - For Vainglory, we chose 32-bit format to avoid artefacts
    - Beware - on some GPUs, this can double blend bandwidth



Rendering quality: Shader precision

- Make use of SPIR-V’s RelaxedPrecision Decoration
  - **GLSL:** lowp & mediump (RelaxedPrecision), highp (empty)
  - Prefer RelaxedPrecision – usually faster & more power efficient
  - Watch out for calculations that require full precision

Rendering quality: Texture compression

	OpenGL ES 2.0 (ETC1)	Vulkan (ASTC)
APK size	599 MB	521 MB
Memory (run-time)	1115 MB	557 MB

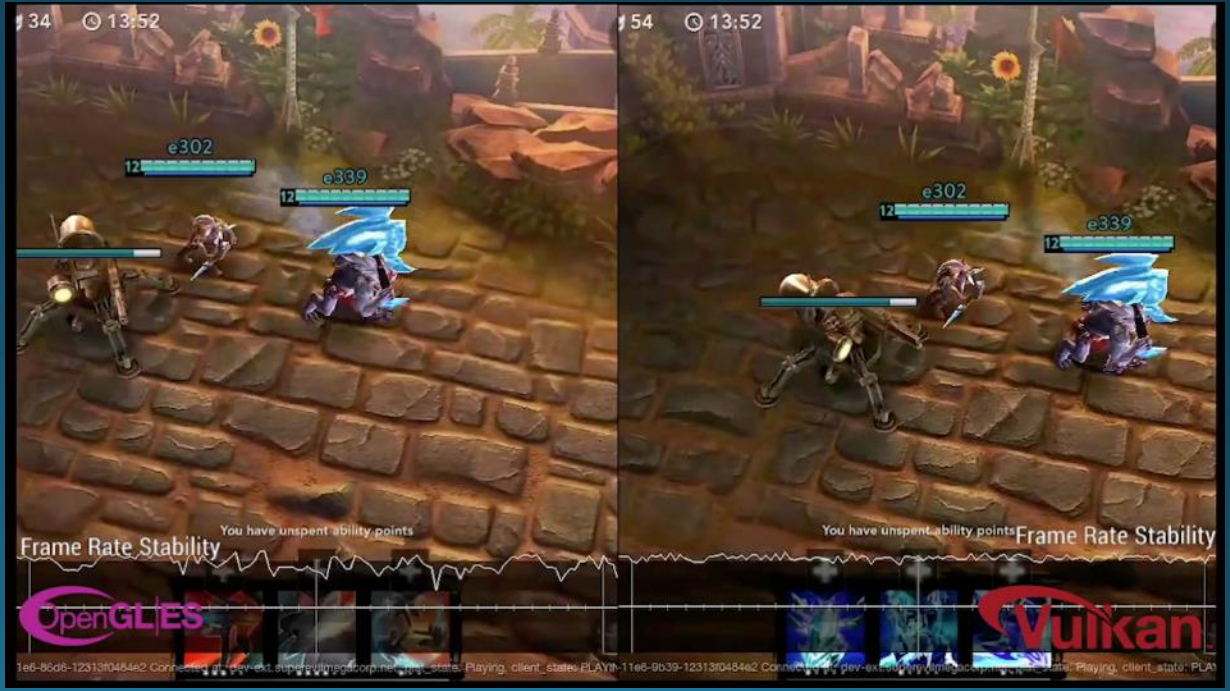
- ETC1
  - Max. three colour channels (RGB)
  - Bit-rates used
    - 4 bits per-pixel
- ASTC
  - Max. four colour channels (RGBA)
  - Bit-rates used
    - 3.56 bits per-pixel (6x6 block size)
    - 2.0 bits per-pixel (8x8 block size)

SAMSUNG

Vainglory : Performance Gain

- Vainglory was already well optimized and still near 60 FPS in GLES
- Vulkan gives everlasting 60 FPS in any case with big reduction in memory usage by using ASTC

Vulkan + ASTC vs GLES + ETC1		
Performance	Normal	4 %
	Throttling	30 %
Power usage		5 %
Memory usage		25 %



VG: Vulkan Beta

- Initial release
  - August 2016
  - One of the first Android Vulkan titles!

- Current status
  - Updated monthly
  - Lots of user feedback & bugs ironed out
  - Significant performance, memory and power saving over GLES with ETC1

SAMSUNG

Want to know more?

Get in touch!

f.garnier@samsung.com

gamedev@samsung.com

@f.garnier_

Bringing Vainglory to Vulkan: **Questions?**

SAMSUNG