



arm

Vulkan Best Practice for Mobile Developers

Vulkanised 2019

Attilio Provenzano
20th May 2019

Vulkan Best Practice For Mobile Developers

arm

+

Runnable samples

+

Tutorials

+

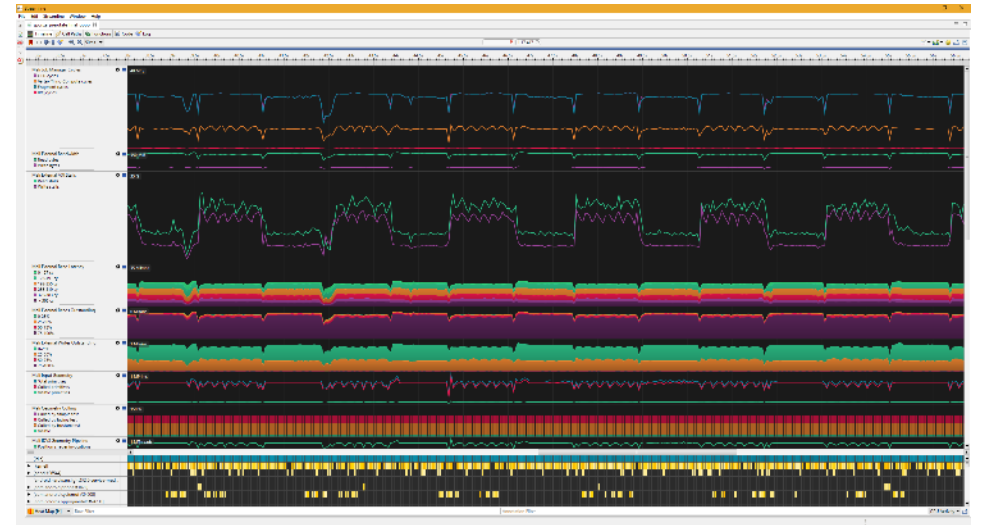
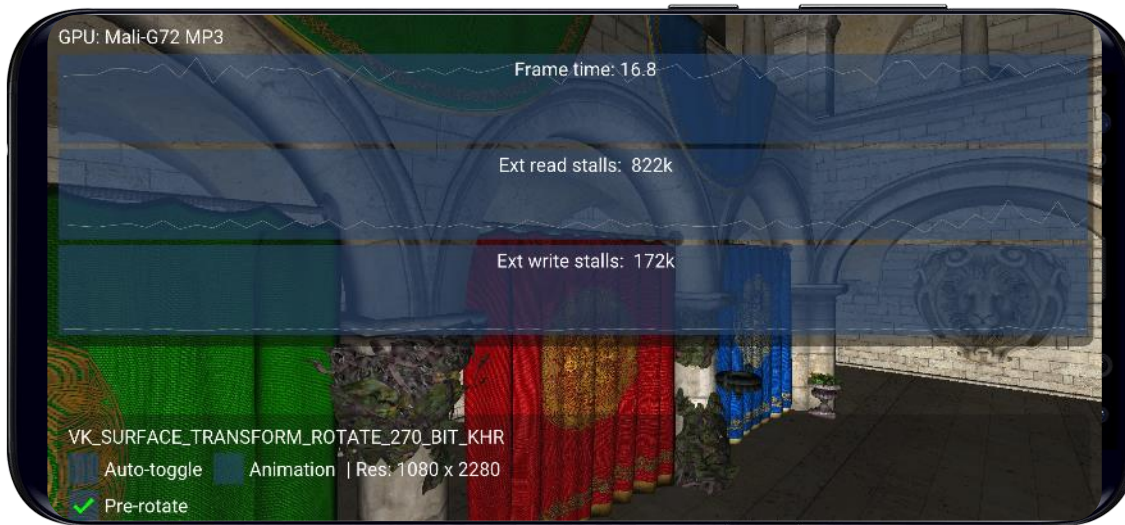
Performance
analysis

+

Mobile-optimized, multi-platform framework

Vulkan best practice for mobile developers

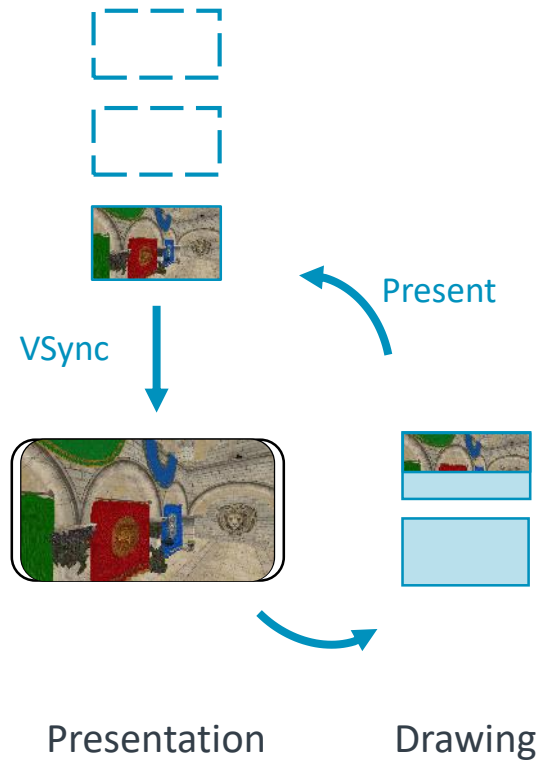
- https://github.com/ARM-software/vulkan_best_practice_for_mobile_developers
- Multi-platform (Android, Windows, Linux)
- Hardware counters displayed on device (no need for root) with HWCPipe
- In-detail explanations, backed-up with data, of best-practice recommendations
- Guide to using performance profiling tools and analysing the results



arm

Sample 1:
N-Buffering

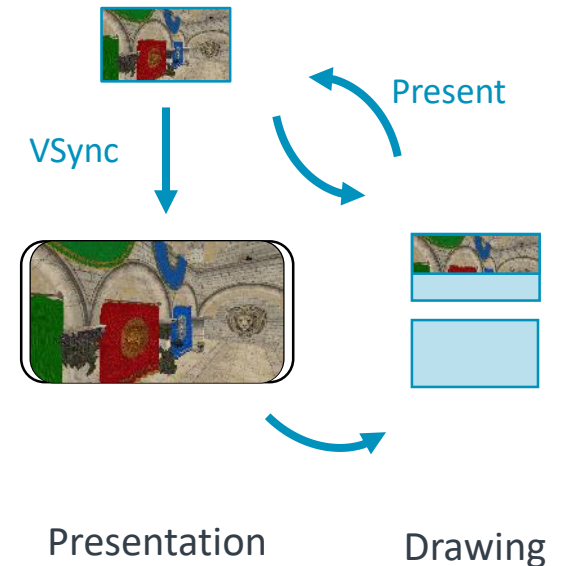
Presentation modes



- + VSync bound
- + Best for mobile
- + Triple buffering

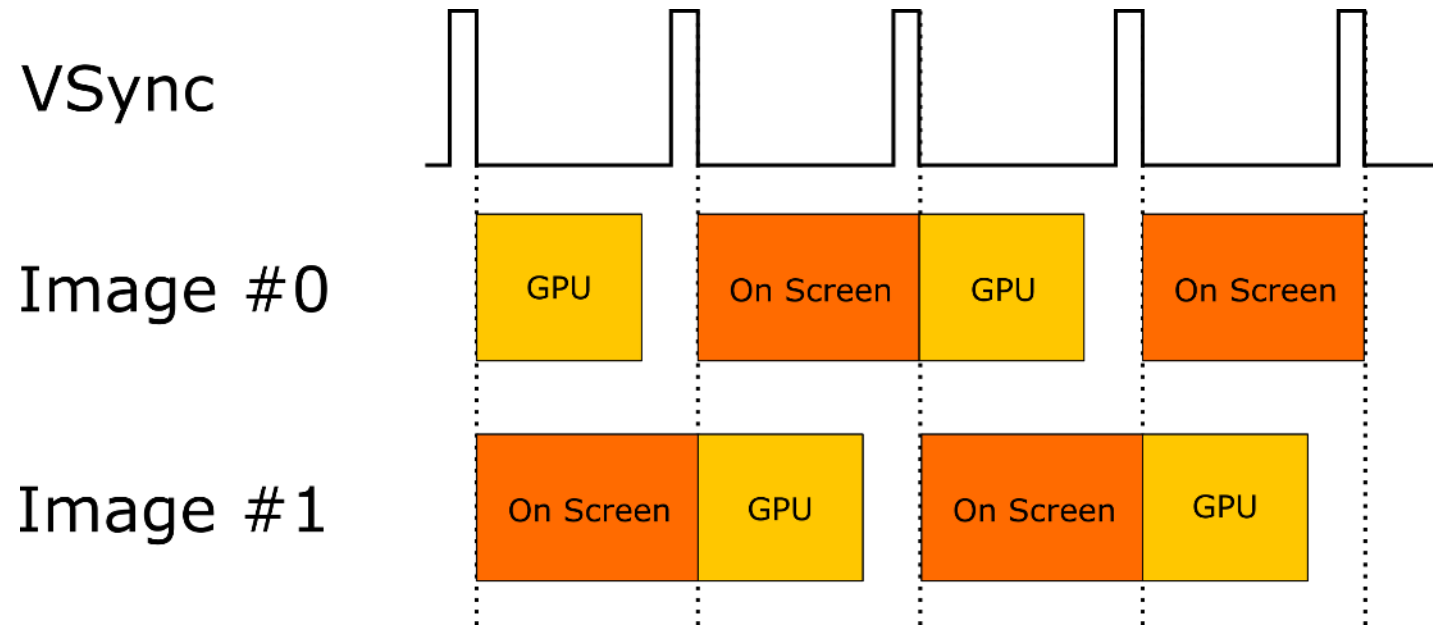


- + Keep submitting
- + Low latency
- + Not optimal for mobile



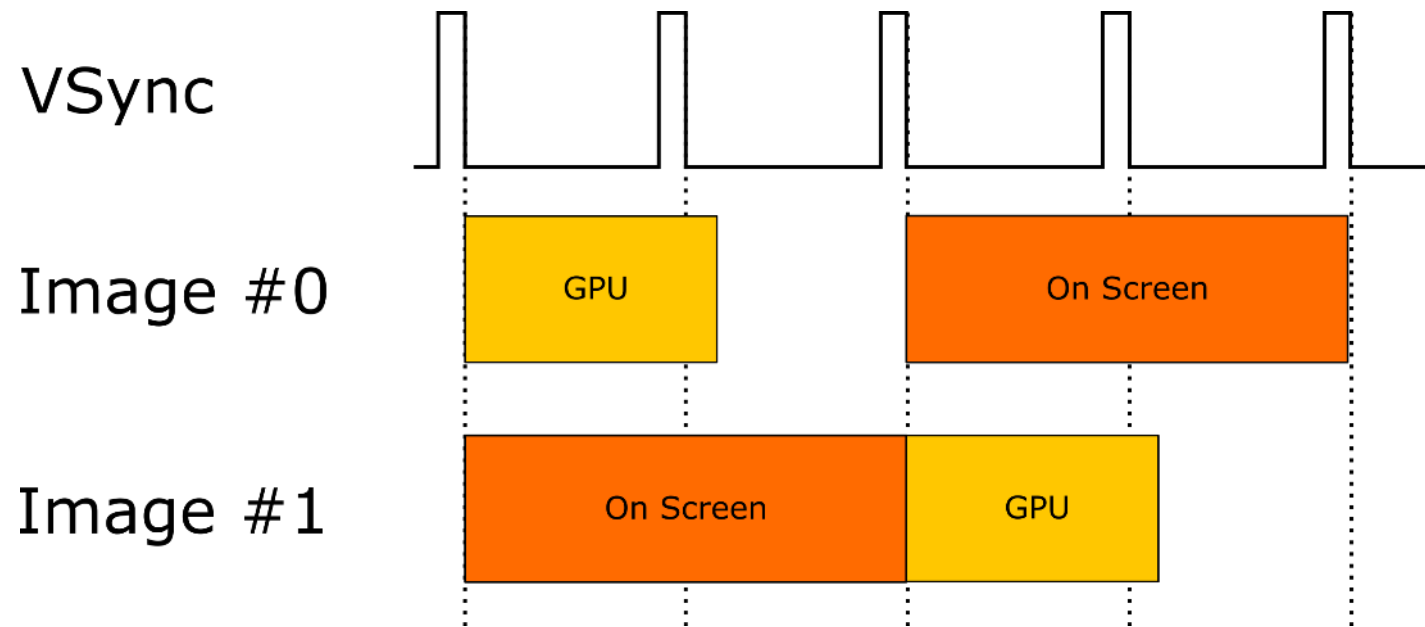
Double buffering

- Double buffering works well if frames can be processed within 16.6 ms
 - At each VSync signal the processed image is presented on screen
 - The previously presented one becomes available to the application again



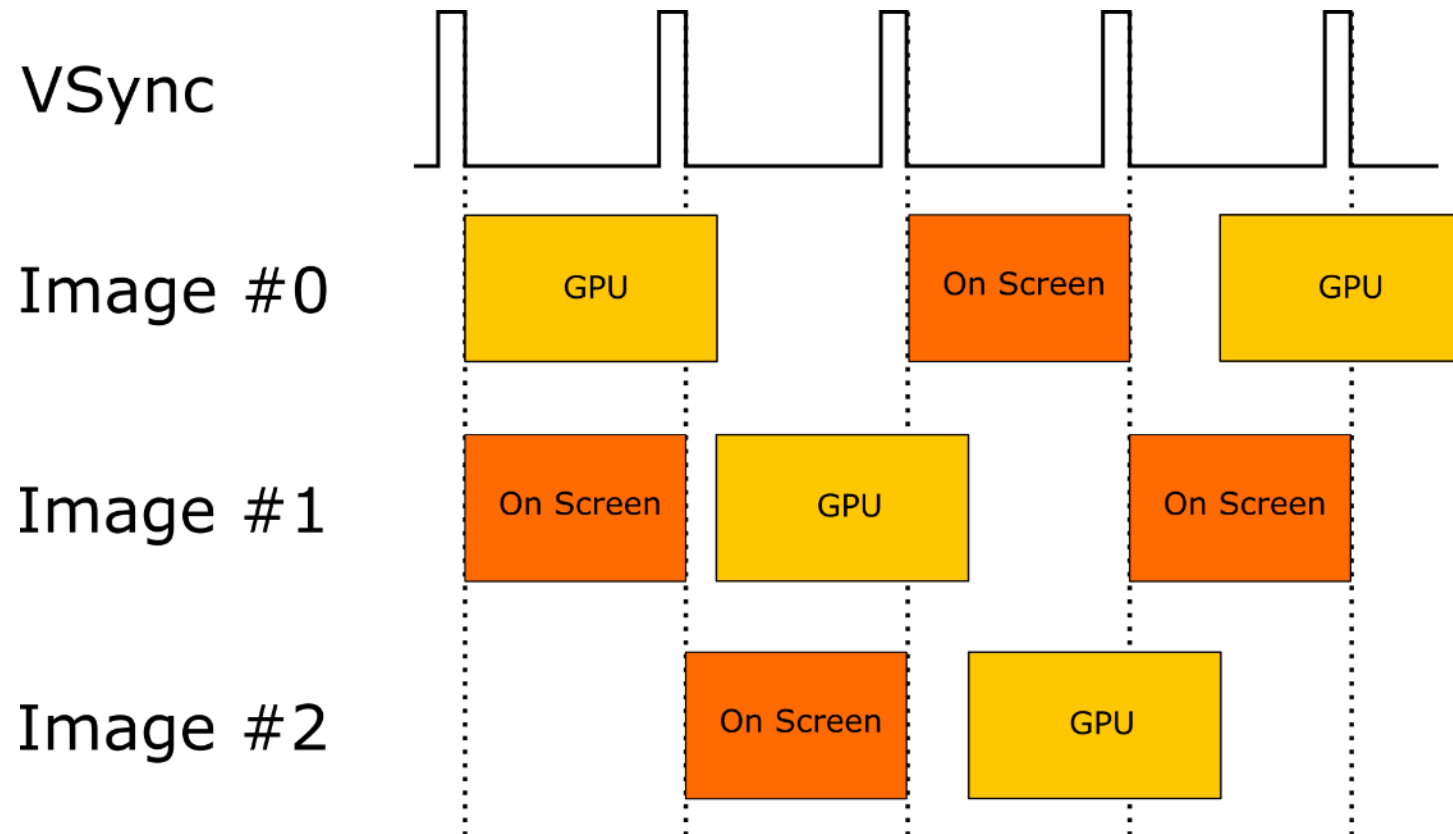
Double vs Triple buffering

- Double buffering breaks if frames take more than 16.6ms
- This idling behaviour caps frame rate at 30fps, while the application could achieve 50



Double vs Triple buffering

- With triple buffering there will always be an image ready for presentation, no stalling



N-Buffering: sample

- The application can ask for a minimum number of images by setting the `minImageCount` parameter in `vkCreateSwapchainKHR`
 - 2 for double buffering
 - 3 for triple buffering
- `VK_PRESENT_MODE_MAILBOX_KHR` might reduce input latency, but it is not optimal for mobile because it keeps the CPU and GPU active while not strictly necessary
- Therefore we recommend `VK_PRESENT_MODE_FIFO_KHR` and `minImageCount=3`

N-Buffering: sample



Up to **x2**
faster frame time

arm

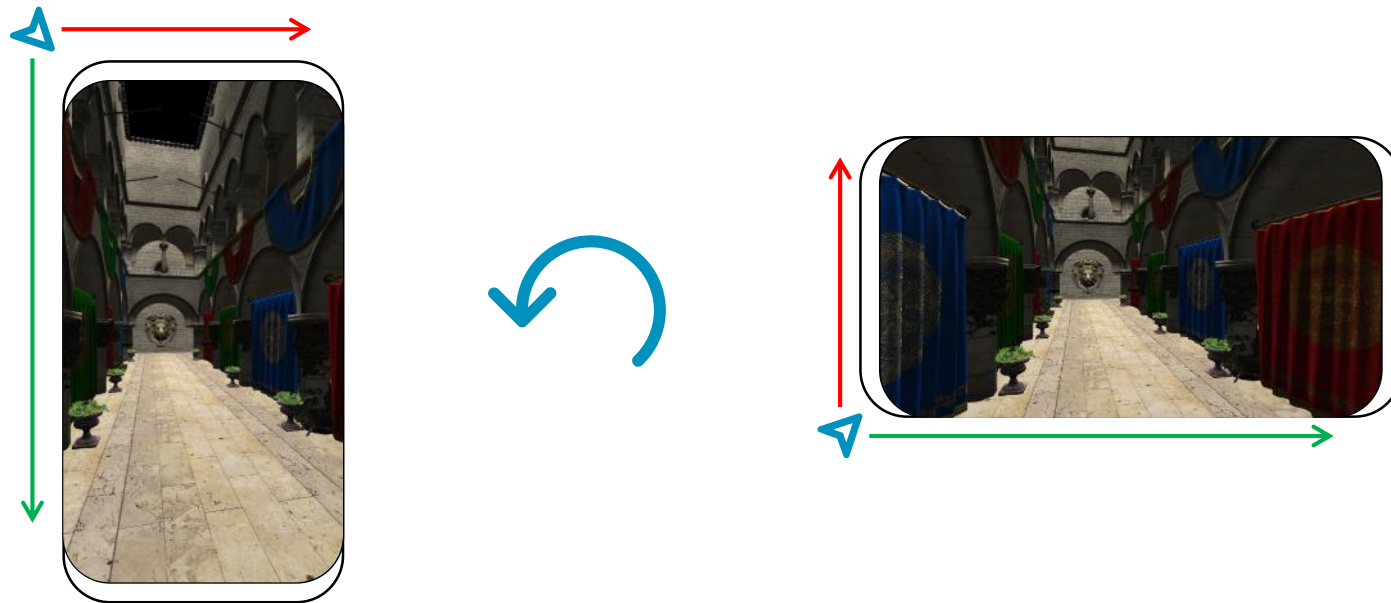
Sample 2:
Pre-rotation

Rotation in mobile devices



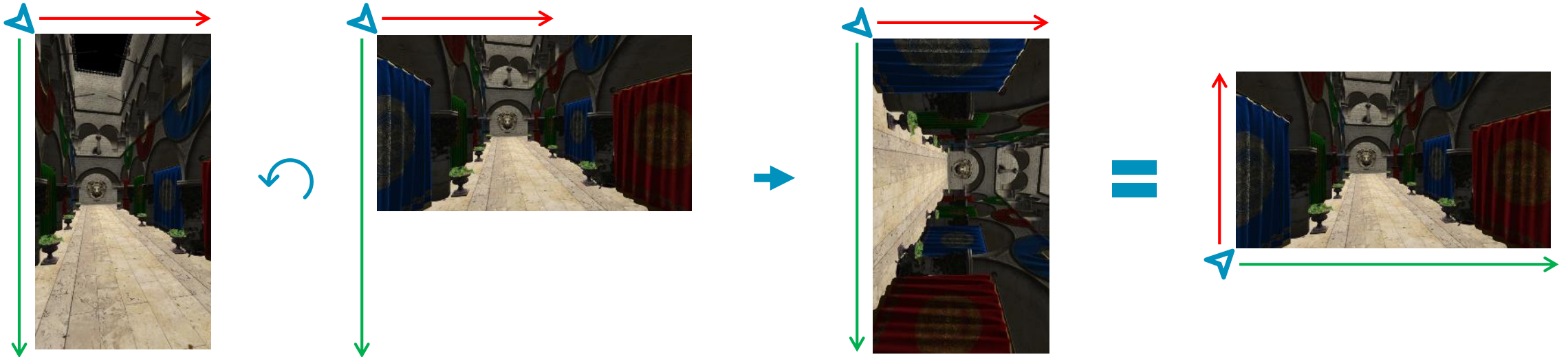
Pre-rotation: theory

- The Display Processor will always draw top to bottom, left to right
- As far as the Display Processor is concerned, nothing changed



Rotation in mobile devices

- Behind the scenes, a change in orientation requires:
 1. An adjusted resolution
 2. A rotation



Pre-rotation

- In OpenGL ES the driver transparently handles this rotation
- In Vulkan, it is the responsibility of the application
- If you rotate the scene after rendering, this extra pass consumes resources
- We recommend you render a rotated scene in the first place: pre-rotation

- No pre-rotation:

1. Draw



2. Rotate



3. Present

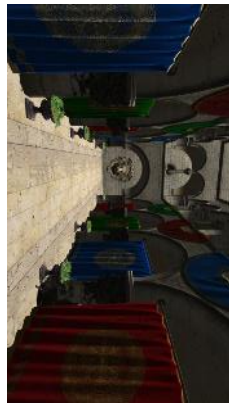


Pre-rotation

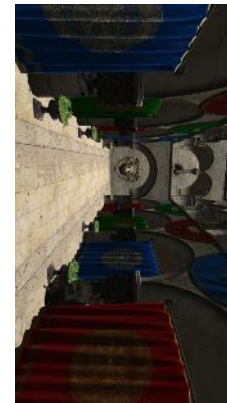
- In OpenGL ES the driver transparently handles this rotation
- In Vulkan, it is the responsibility of the application
- If you rotate the scene after rendering, this extra pass consumes resources
- We recommend you render a rotated scene in the first place: pre-rotation

- With pre-rotation:

1. Draw



2. Rotate



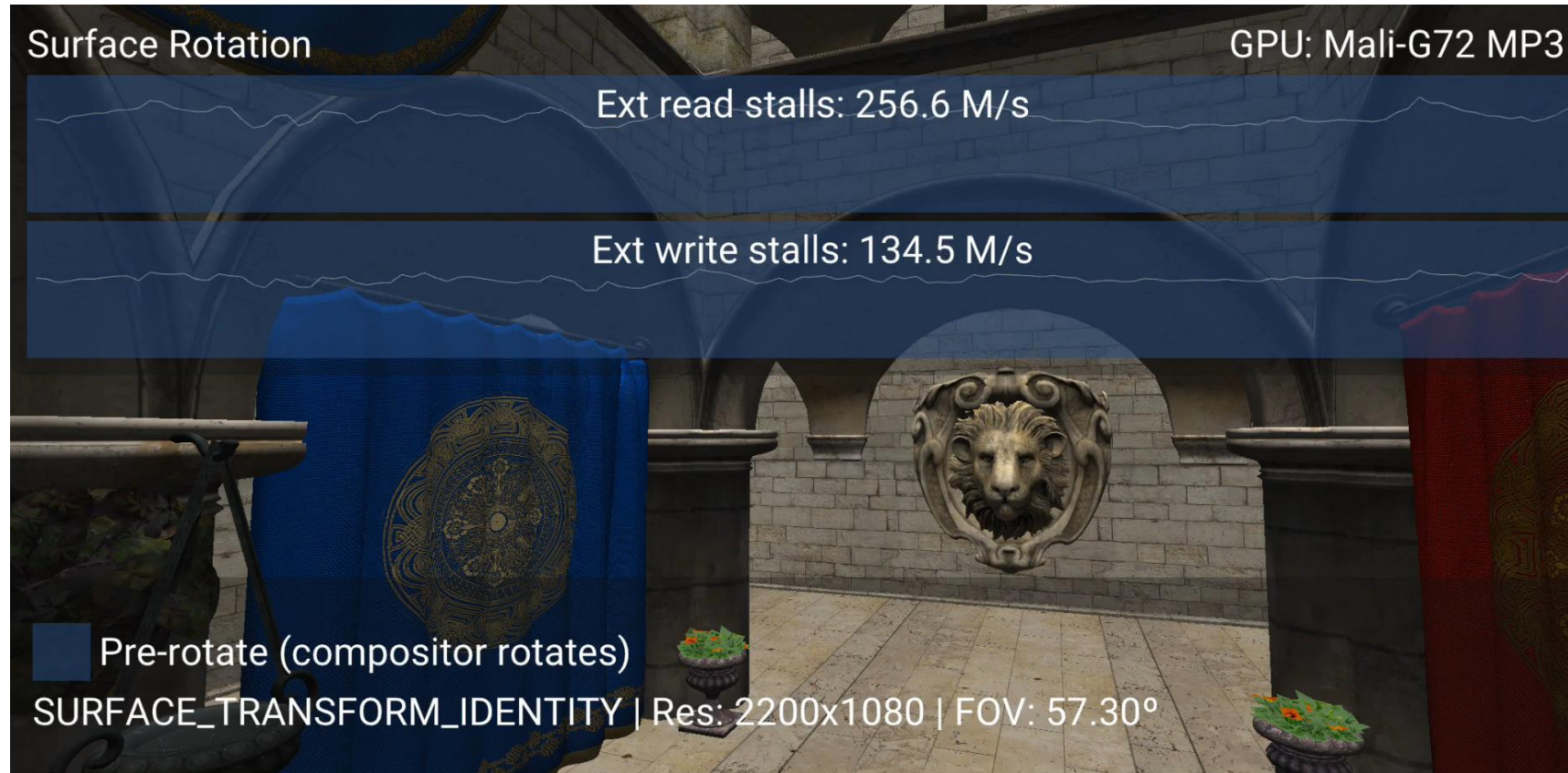
3. Present



Pre-rotation: sample

- On rotation, use `vkGetPhysicalDeviceSurfaceCapabilitiesKHR` to query:
 - `currentExtent`
 - `currentTransform` e.g. `VK_SURFACE_TRANSFORM_ROTATE_90_BIT_KHR`
- Re-create the swapchain ensuring that `preTransform` matches `currentTransform`
- This communicates that the application is handling the rotation, and no extra passes are needed, saving performance
- Do not change the images dimensions, instead draw a rotated version of the world


Pre-rotation: sample



* Screen recording reduces the benefits to 27% and 47%

Up to **88%**
savings in external read stalls

Up to **91%**
savings in external write stalls


Only applicable in devices with
no DPU rotation support

arm

Sample 3:
Load/Store
operations

Load operations

- `loadOp` operations define how to initialize memory at the start of a render pass

+ `LOAD_OP_LOAD`

+ `LOAD_OP_CLEAR`

+ `LOAD_OP_DONT_CARE`

- Clear or invalidate each attachment at the start of a render pass using `LOAD_OP_CLEAR` or `LOAD_OP_DONT_CARE`

Store operations

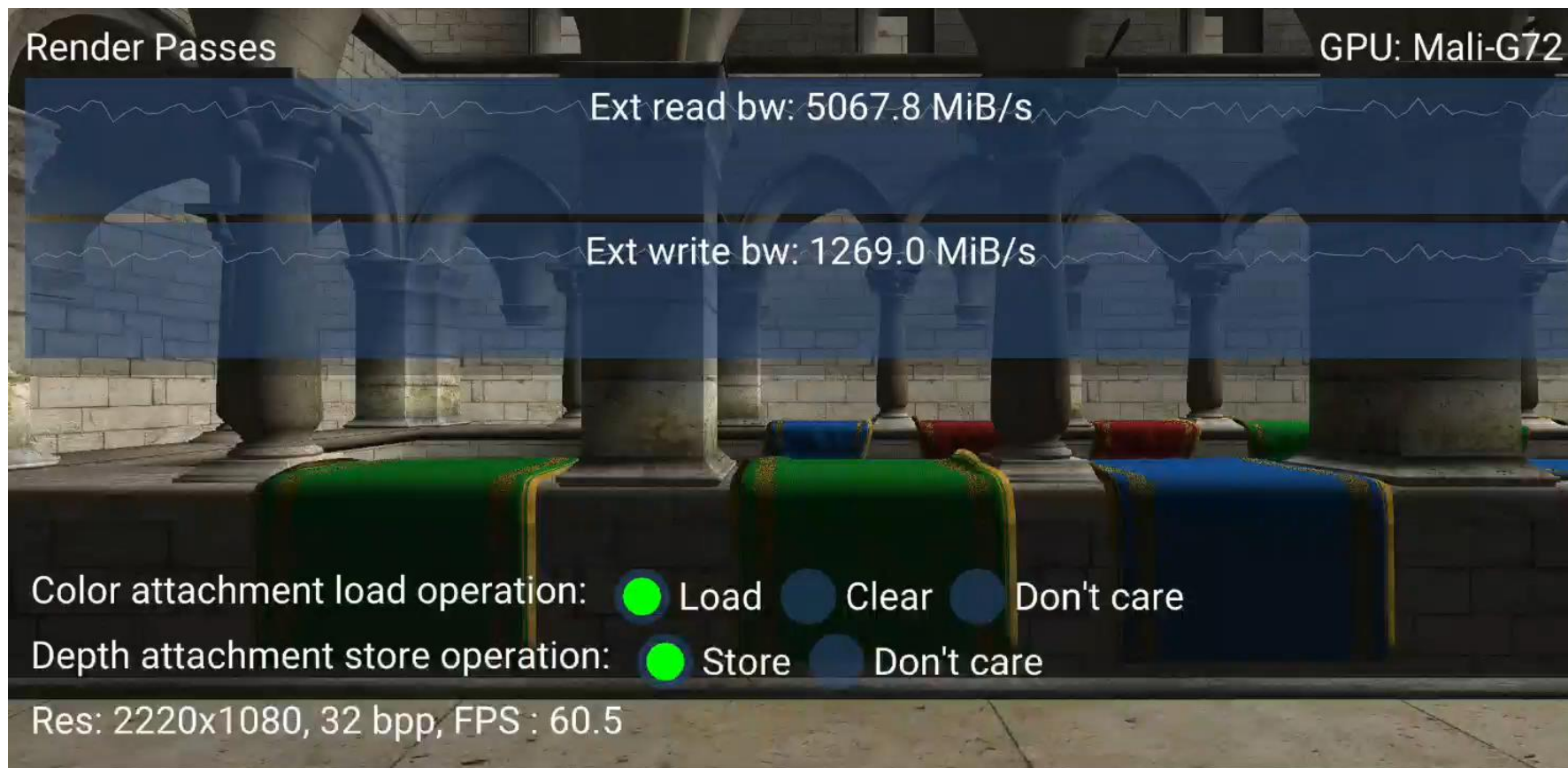
- `storeOp` operations define what is written back to main memory at the end of a pass

✦ `STORE_OP_STORE`

✦ `STORE_OP_DONT_CARE`

- If they are not going to be used further, ensure that the contents are invalidated at the end of the render pass using `STORE_OP_DONT_CARE`

Load/Store operations: sample



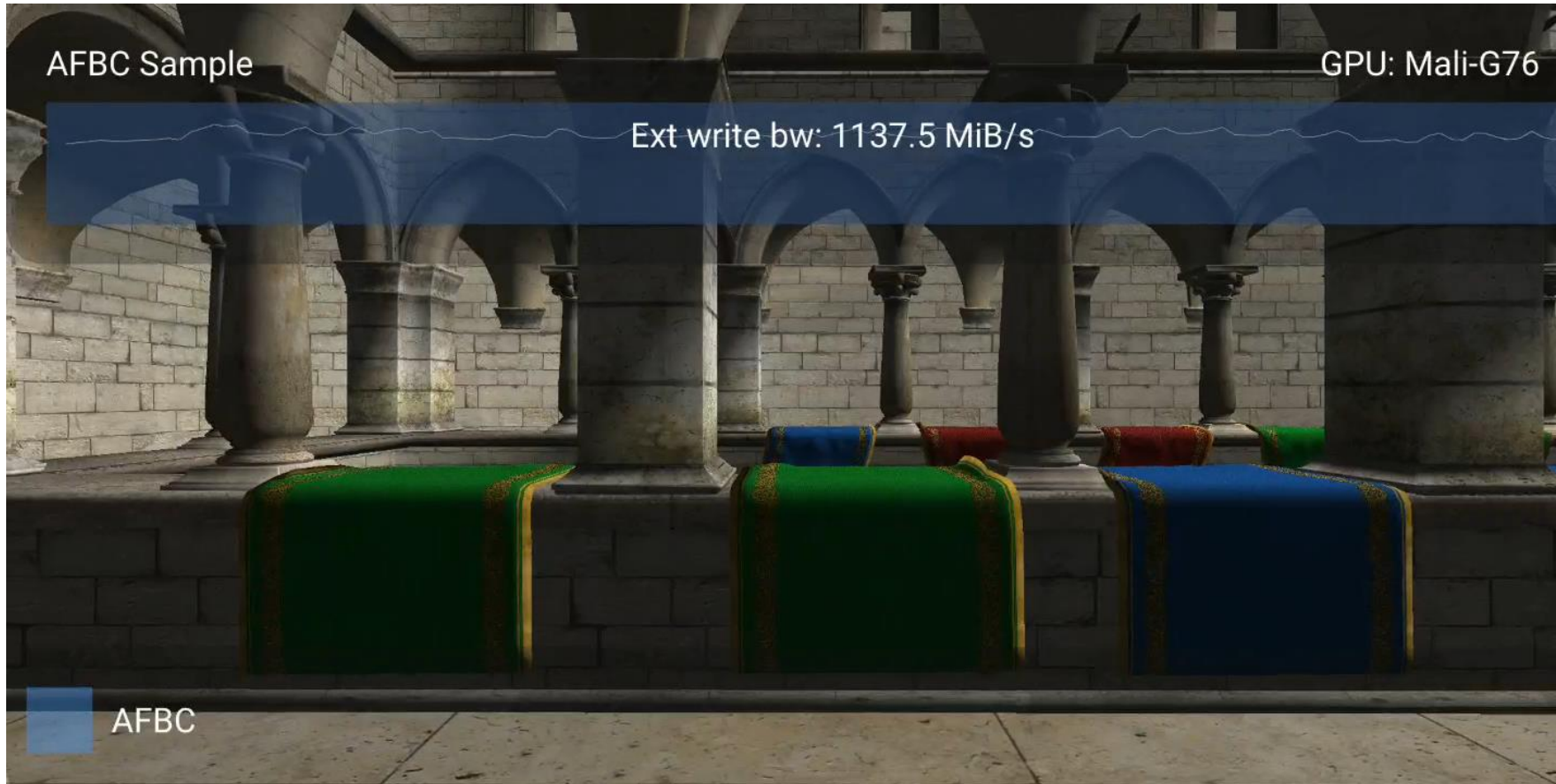
Up to **12%**
savings in external read cycles

Up to **50%**
savings in external write cycles

arm

Sample 4:
AFBC

Arm Framebuffer Compression (AFBC)



Up to **33%**
savings in external write cycles

arm

Framework

Framework

- Platform independent (Android, Linux and Windows)
- Maintain close relationship with Vulkan objects
- Runtime GLSL shader variant generation + shader reflection (Khronos' SPIRV-Cross)
 - Simplify creation of Vulkan objects:
 1. VkRenderPass
 2. VkFramebuffer
 3. VkPipelineLayout
 4. VkDescriptorSetLayout
- Load 3D models (glTF 2.0 format)
 - Internal scene graph

Initialization

Render Pass

Attachment Description

Input Attachment

Output Attachment

Descriptor Set Layout

Texture Binding

Uniform Binding

Descriptor Pool

Texture Binding

Uniform Binding

Framebuffer

Render Pass

Image View

Image View

Pipeline Layout

Descriptor Set Layout

Push constants

Descriptor Set

Descriptor Set Layout

Descriptor Pool

Image View

Image View

Graphics Pipeline

Render Pass

Pipeline Layout

Subpass

Shader module

Shader module

Vertex Input

Input Assembly

Rasterization

Multisample

Depth Stencil

Color Blend

Shader Module

Texture Resource

Image Resource

Input Resource

Output Resource

Object/Dependency

Application defined

Initialization

Render Pass

Attachment Description

Input Attachment

Output Attachment

Descriptor Set Layout

Texture Binding

Uniform Binding

Descriptor Pool

Texture Binding

Uniform Binding

Framebuffer

Render Pass

Image View

Image View

Pipeline Layout

Descriptor Set Layout

Push constants

Descriptor Set

Descriptor Set Layout

Descriptor Pool

Image View

Image View

Graphics Pipeline

Render Pass

Pipeline Layout

Subpass

Shader module

Shader module

Vertex Input

Input Assembly

Rasterization

Multisample

Depth Stencil

Color Blend

Shader Module

Texture Resource

Image Resource

Input Resource

Output Resource

GLSL Compiler

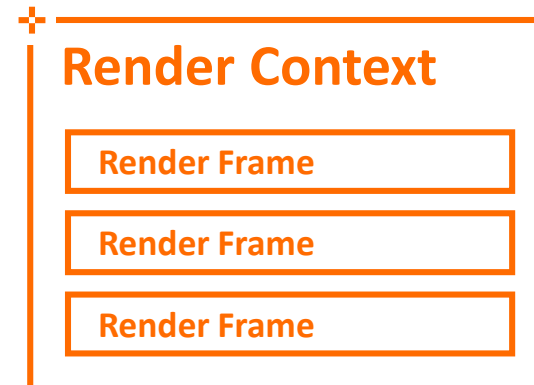
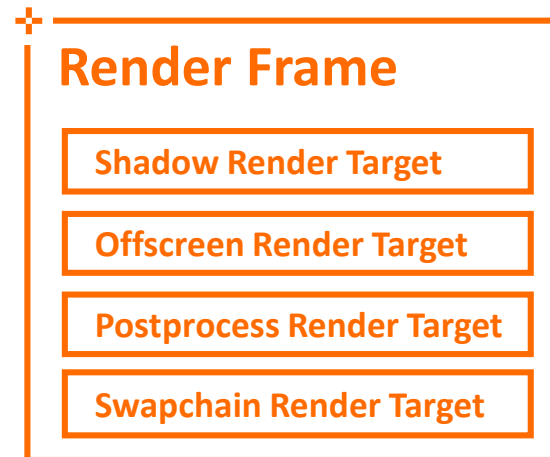
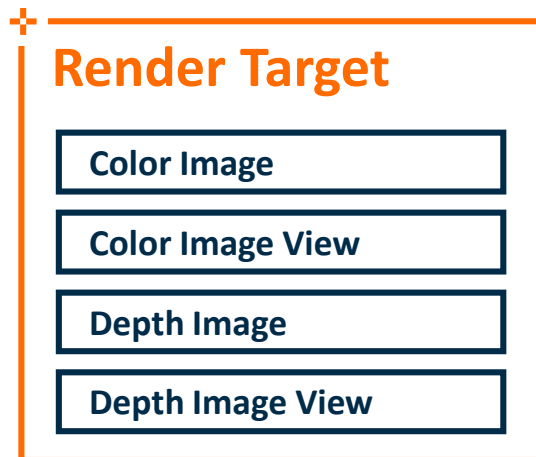
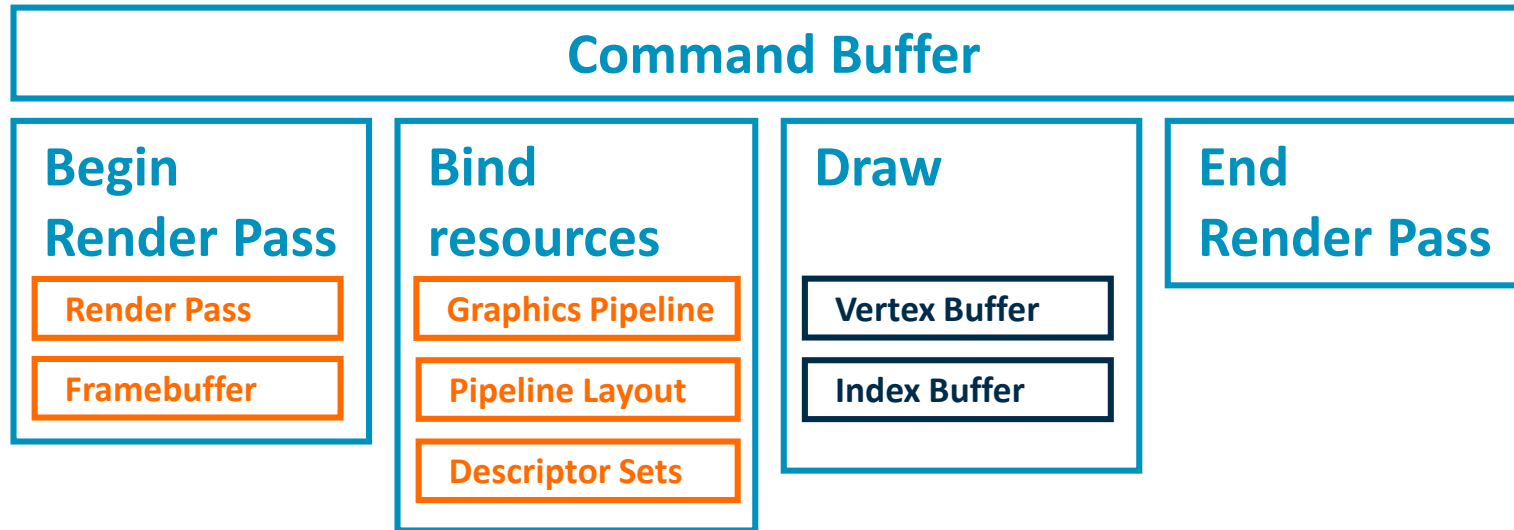
SPIRV Reflection

Generated resource

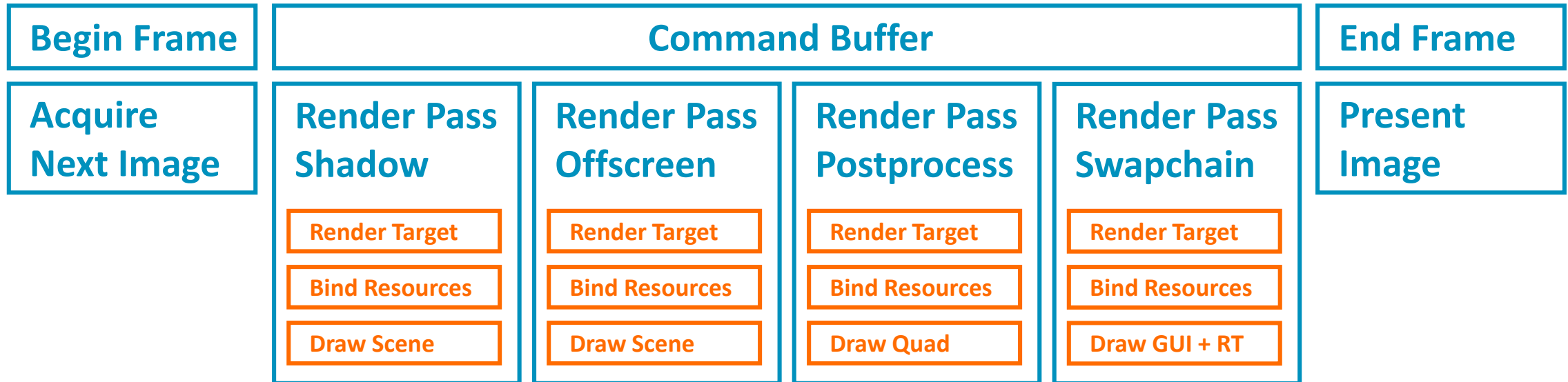
Object/Dependency

Application defined

High-Level API



High-Level API



arm

What's new

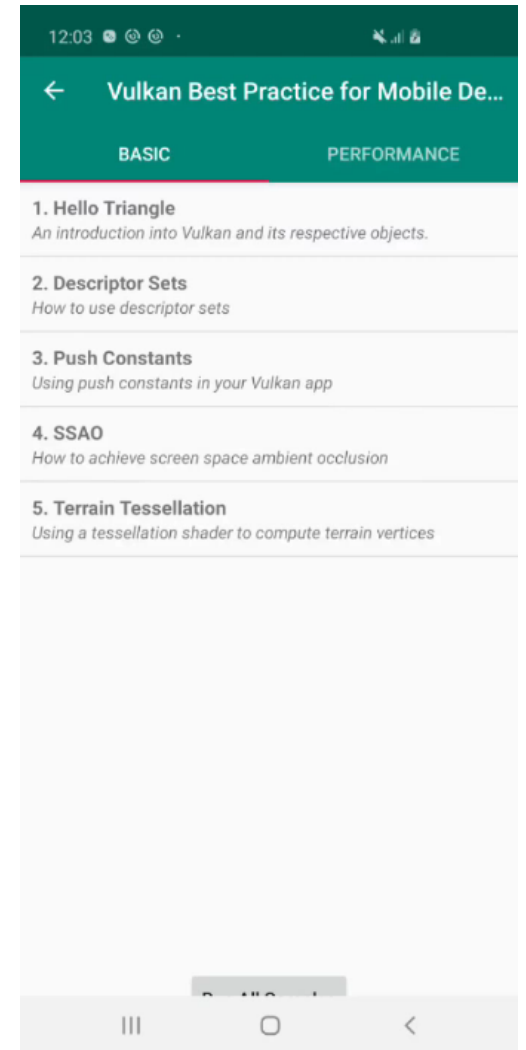
General improvements

- Texture compression
 - Support ASTC with mipmaps (fast decompression on desktop)
 - Support KTX
- More scenes
- Filesystem
- Debug window



Integrating Sascha Willems's samples

- Proof of concept
- Wrapped into our Sample class for the launcher
- Aim to maintain the integrity of the samples



Better profiling

- Platform-independent interface for HWCPipe
 - CpuProfiler and GpuProfiler with counter definitions
 - <https://github.com/ARM-software/HWCPipe>
- Counter sampling with 1 ms resolution
- Specify counters via code or via JSON string

```
// Begin profiling session
```

```
auto h = HWCPipe({CpuCounter::Cycles, CpuCounter::Instructions});  
h.run();
```

```
// Sample counters
```

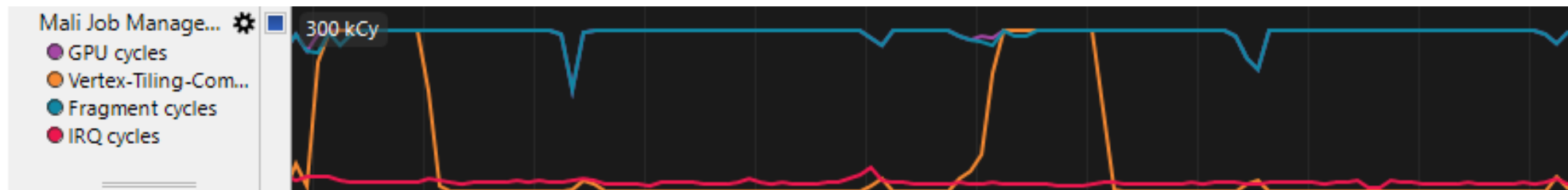
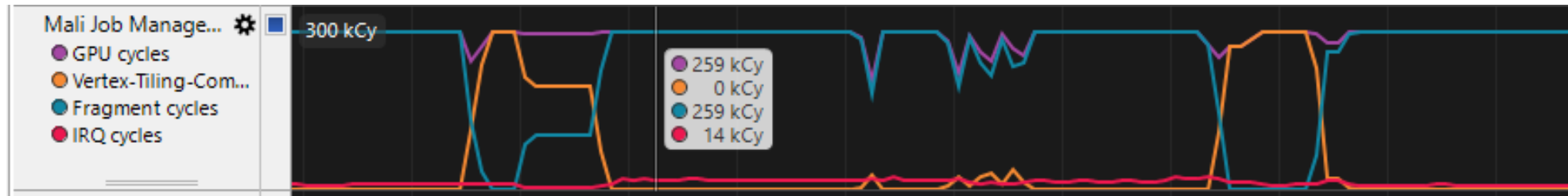
```
auto s = h.sample();  
if (s.cpu) {  
    auto value = s.cpu->at(CpuCounter::Cycles).get<float>();  
}
```

arm

What's next

Samples in flight

- Pipeline caching
- Specialization constants vs uniform buffers
- Workload synchronisation and pipeline barriers



Next samples

- Roadmap on GitHub

+ **Command
buffer reuse**

+ **Multithreaded
rendering**

+ **Deferred
rendering**

+ **Descriptor
management**

- Feedback and contributions welcome!
- https://github.com/ARM-software/vulkan_best_practice_for_mobile_developers

arm

Thank You

Danke

Merci

谢谢

ありがとう

Gracias

Kiitos

감사합니다

धन्यवाद

شكرًا

תודה



The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks