How to Get the Most Out of Mobile VR in Unity

ARM

Roberto Lopez Mendez
Senior Engineer

Vision VR/AR Summit 2017
05/01/2017

Agenda

- New mobile VR features in Unity 5.6
  - Daydream and Google Cardboard native integration
  - Single-pass stereo rendering (multiview)
- Mali Graphics Debugger integration in Unity
  - Live demo
- Mobile VR best practice
  - MSAA, ASTC, optimized rendering techniques based on local cubemaps
- Expected benefits from Vulkan in VR

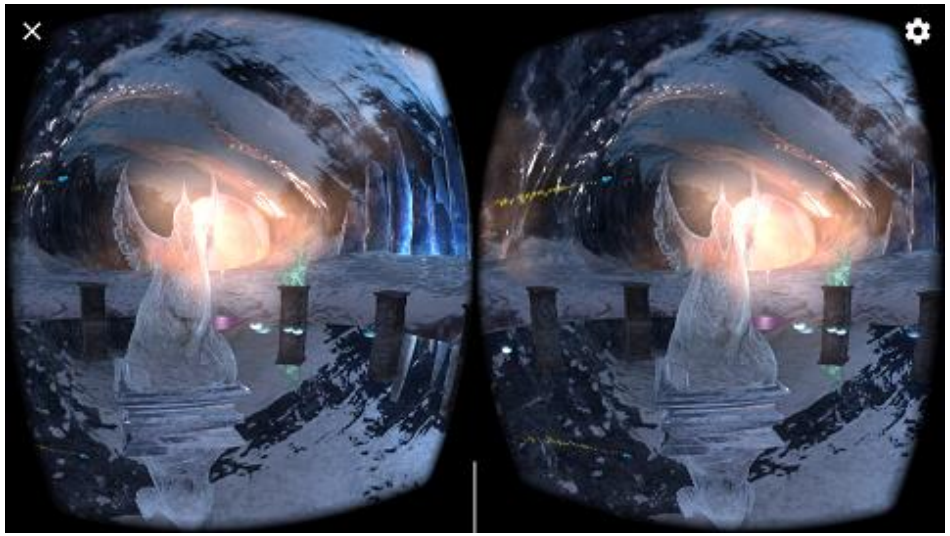
Daydream native integration

Daydream native integration

unity VR

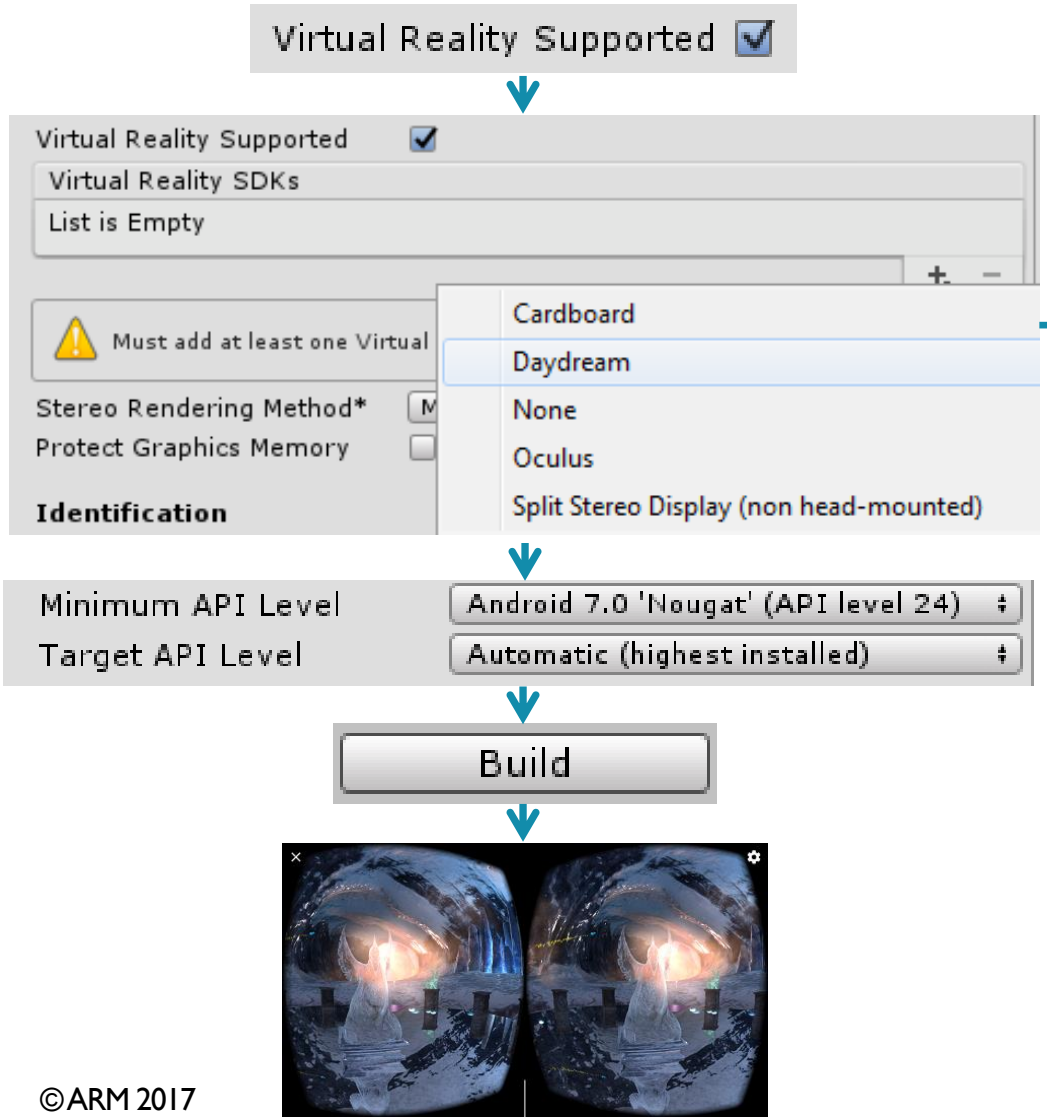
VR Plugin support limitations:

- Each VR device has a different plugin
- Plugins may conflict with each other
- Each release of newer VR SDKs / Runtimes can break older games
- Lower level engine optimizations are not possible with plugin approach of two separate cameras

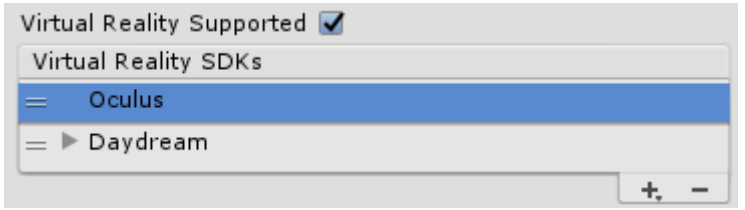


Unity 5.6
Daydream Native Support → Simpler, easier, more efficient and performant

Porting your app to Daydream



Multiple VR SDKs support



- Devices initialized and enabled in the same order as they are listed
- Check current loaded VR device: VRSettings.loadedDeviceName

Daydream controller not yet integrated in Unity 5.6.

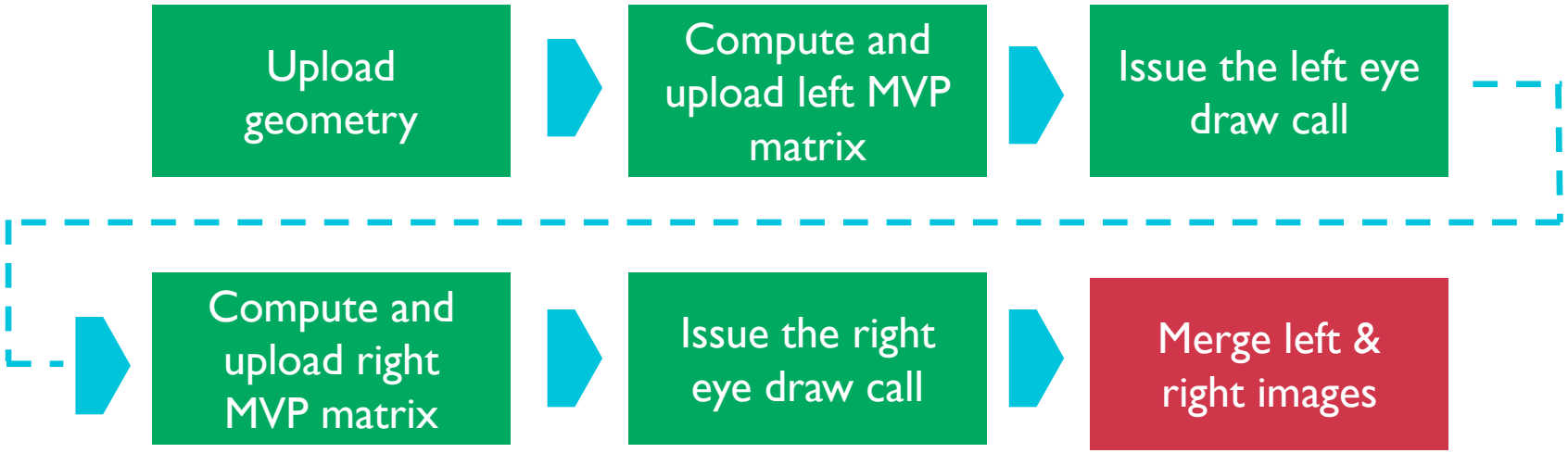
- Add the Google VR SDK to the project
- Add GvrEventSystem and GvrControllerMain prefabs to your scene
- Use GvrController methods to check for user interaction:
  - AppButton, AppButtonUp, AppButtonDown, ClickButton, ClickButtonUp, ClickButtonDown, isTouching, etc.

ARM

Single-pass stereo rendering

Multi-pass vs single-pass stereo rendering

Traditional multi-pass pipeline for rendering stereo images



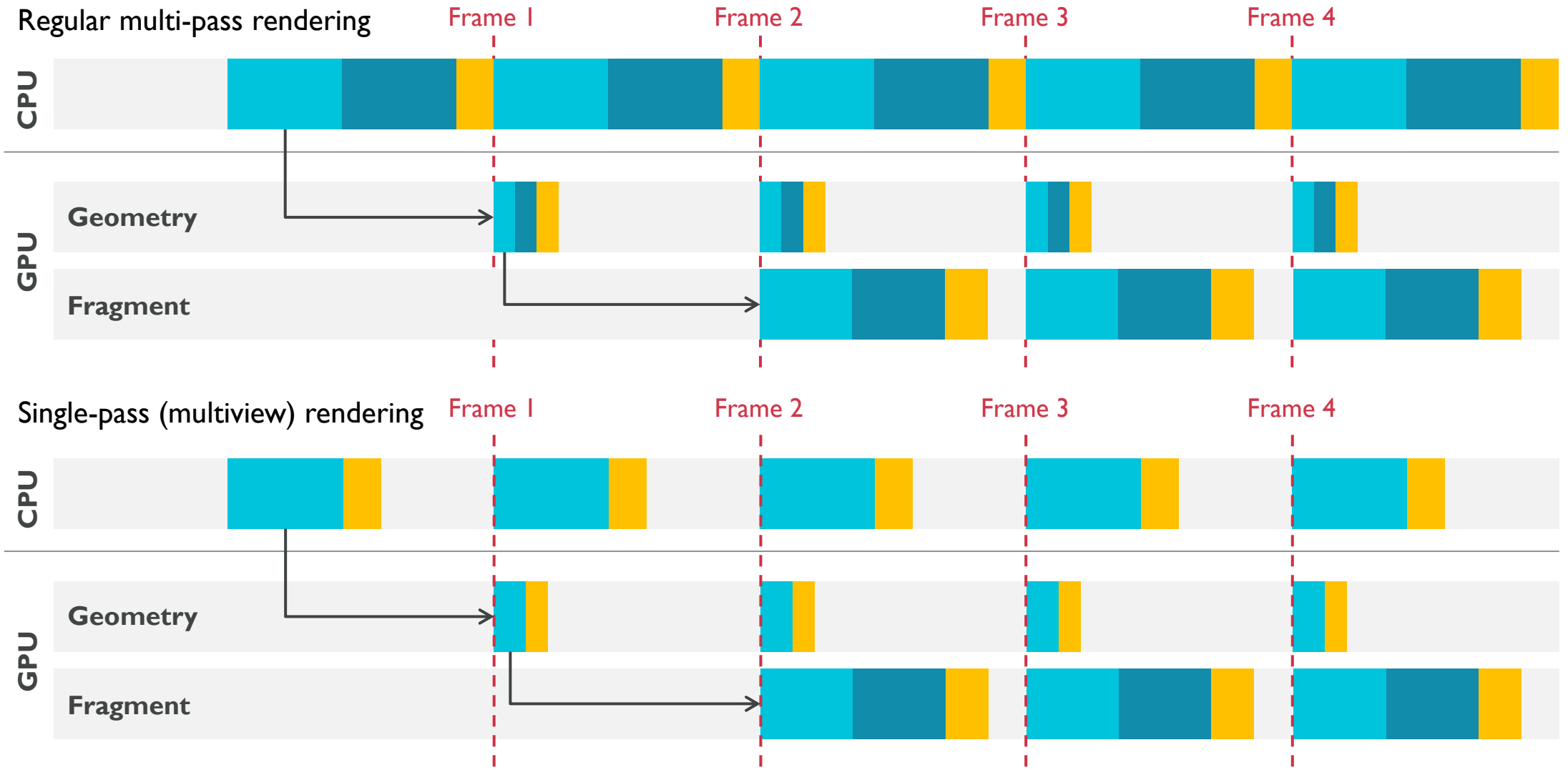
Single-pass (multiview) pipeline for rendering stereo images

Vertex shader with single-pass (multiview)

```
#version 300 es
#extension GL_OVR_multiview2 : enable
precision highp float;
layout(num_views = 2) in;

in vec3 vertexPosition;
in vec2 UVCoordinates;
out vec2 texCoord;

uniform mat4 MVP[2];

void main(){
        gl_Position = MVP[gl_ViewID_OVR] * vec4(vertexPosition, 1.0f);
        texCoord = UVCoordinates;
}
```

← **This line is executed N view times**

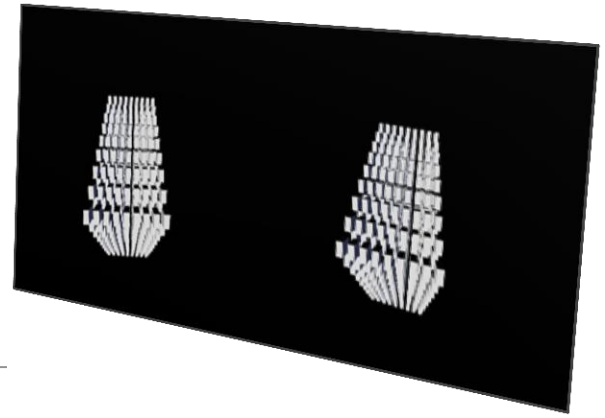
Multi-pass vs single-pass CPU-GPU timeline

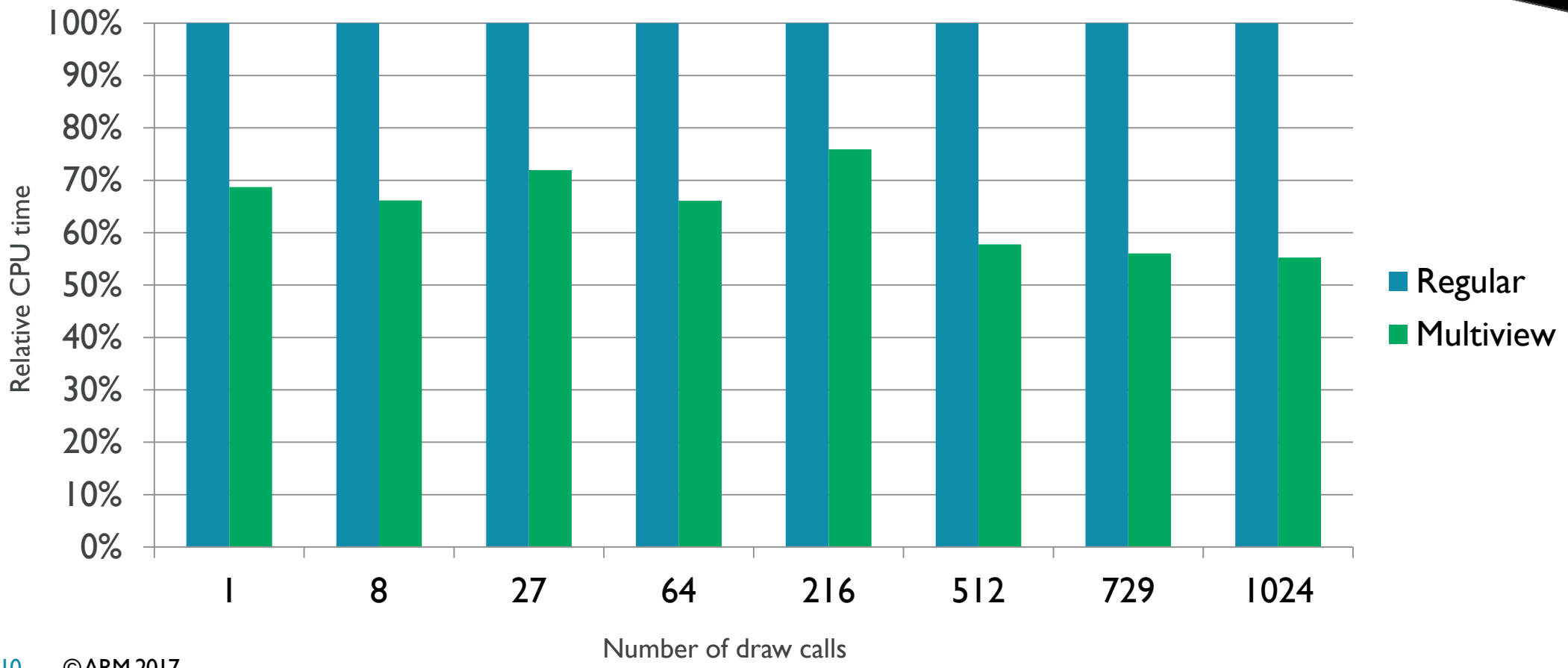


Left eye rendering Right eye rendering Side by side composition

ARM

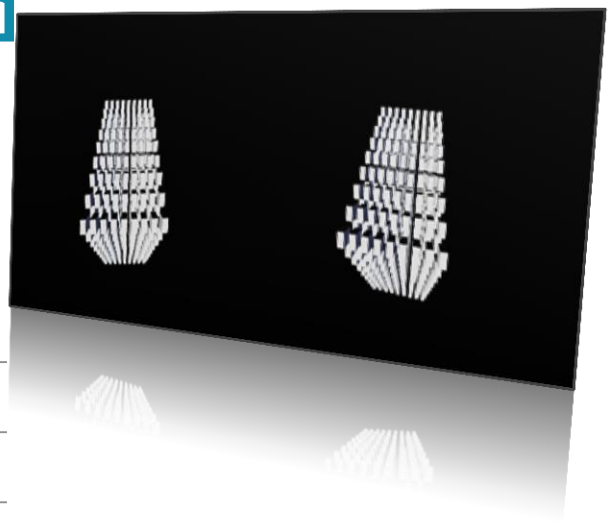
Single-pass (multiview): CPU load

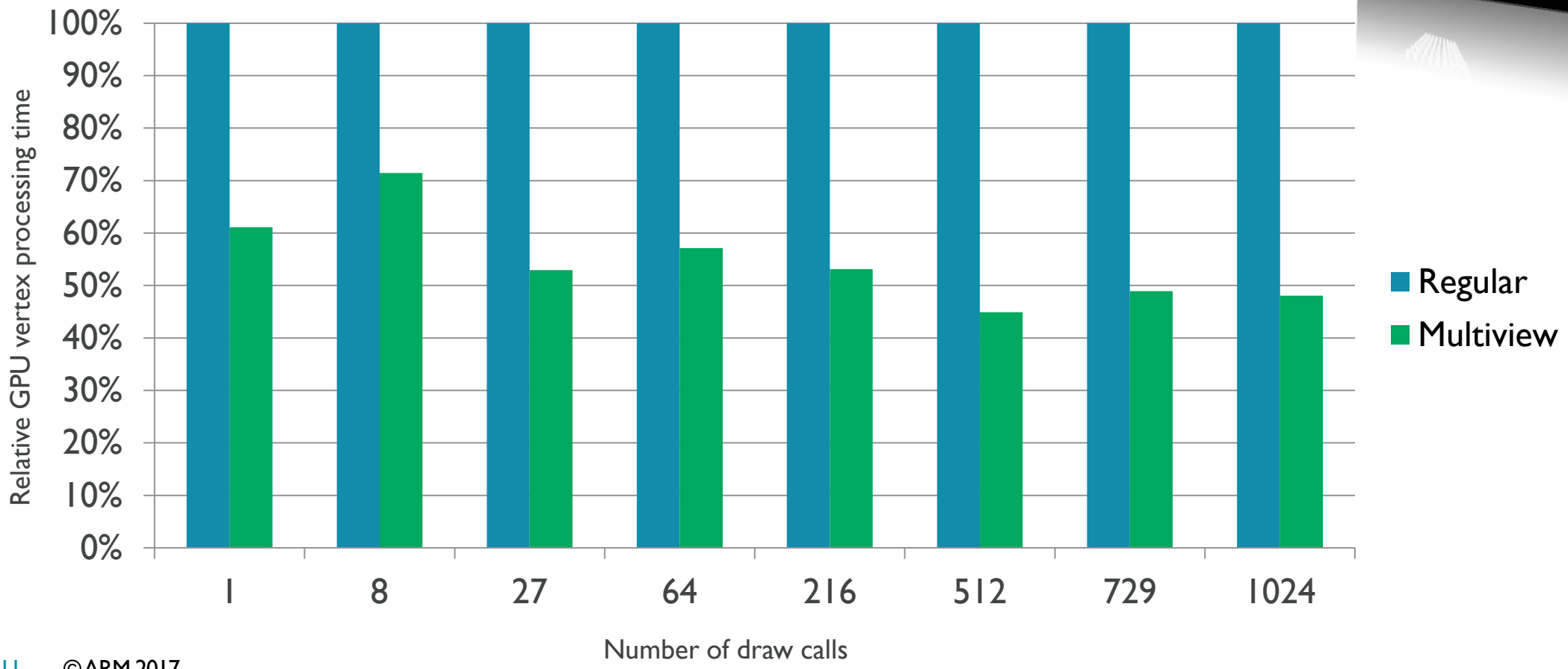




ARM

Single pass (multiview): GPU vertex load



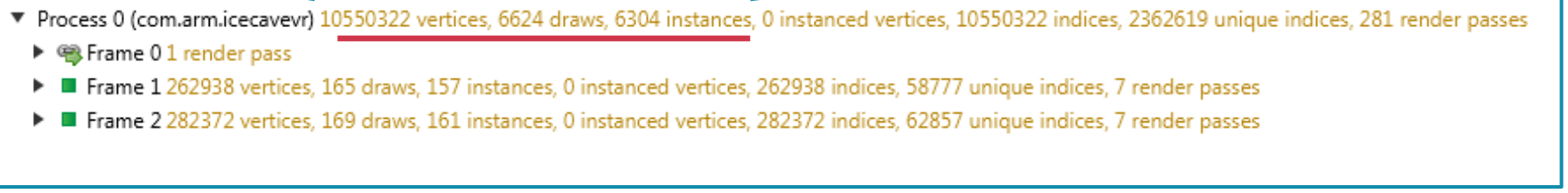


ARM

Multi-pass vs single-pass (Ice Cave VR MGD stats)

Multi-pass

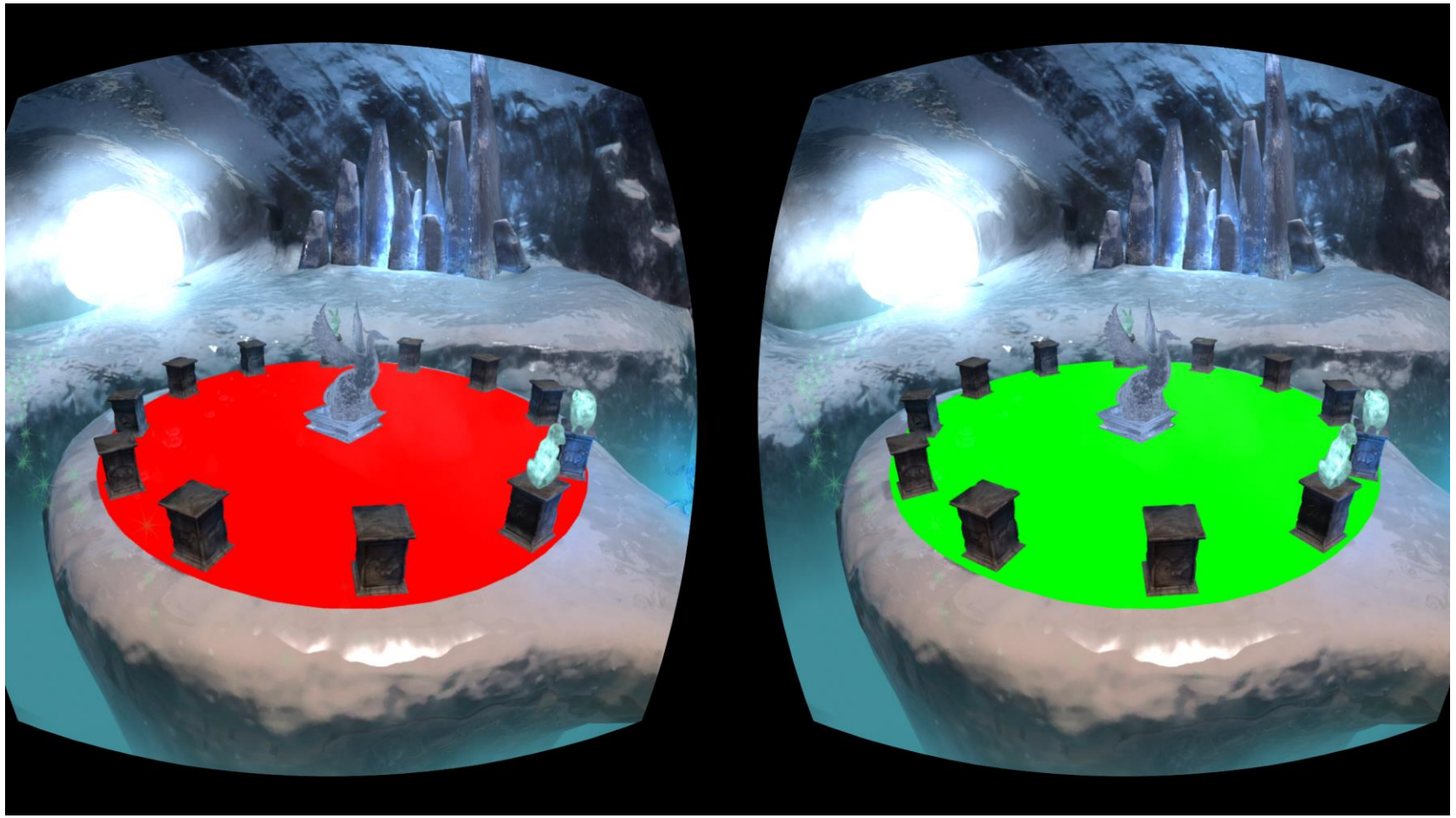
10550 322 vertices, 6624 draws, 6304 instances…



Single-pass

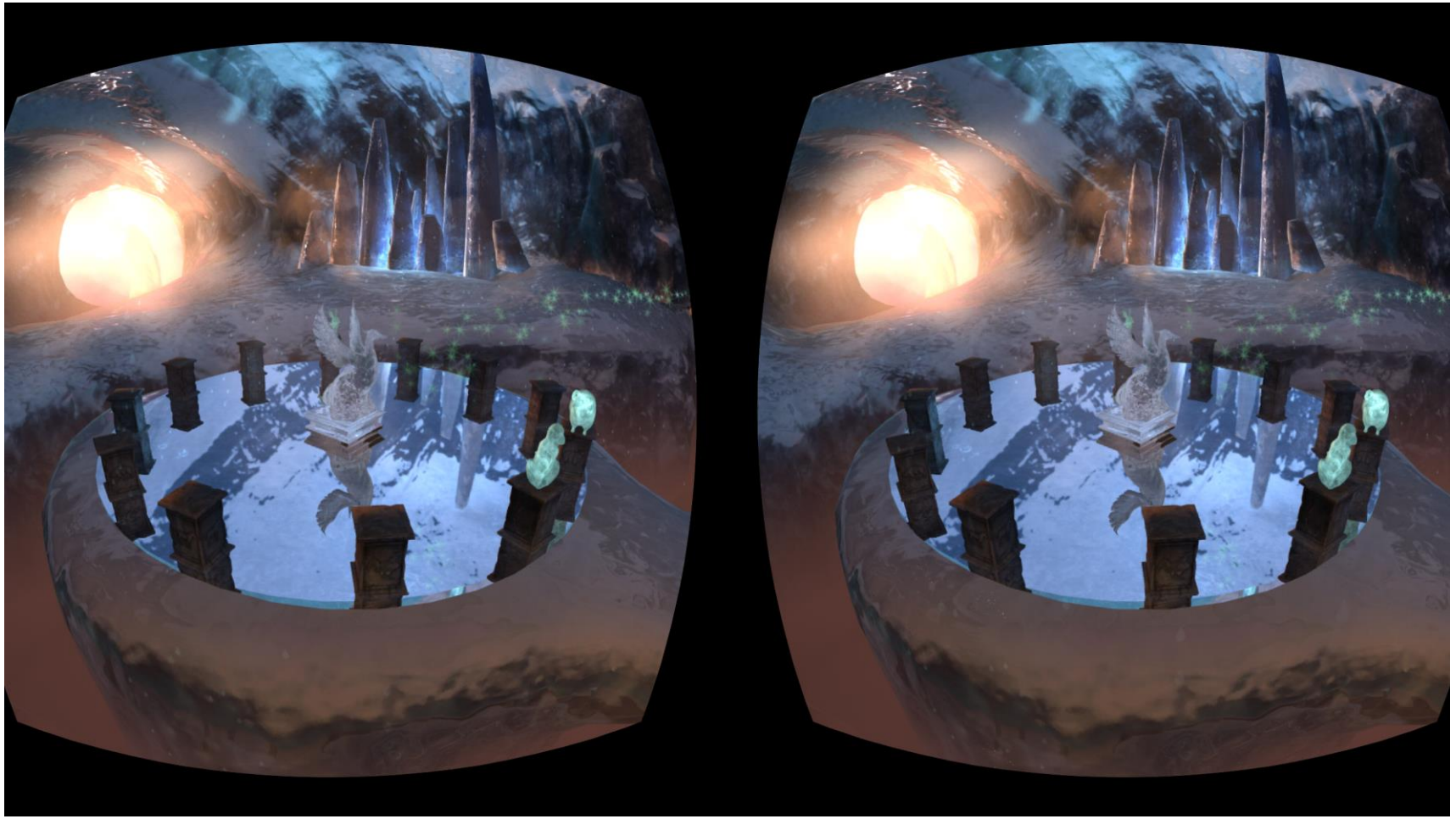
6064536 vertices, 3560 draws, 3350 instances…

Single-pass stereo rendering: stereo reflections

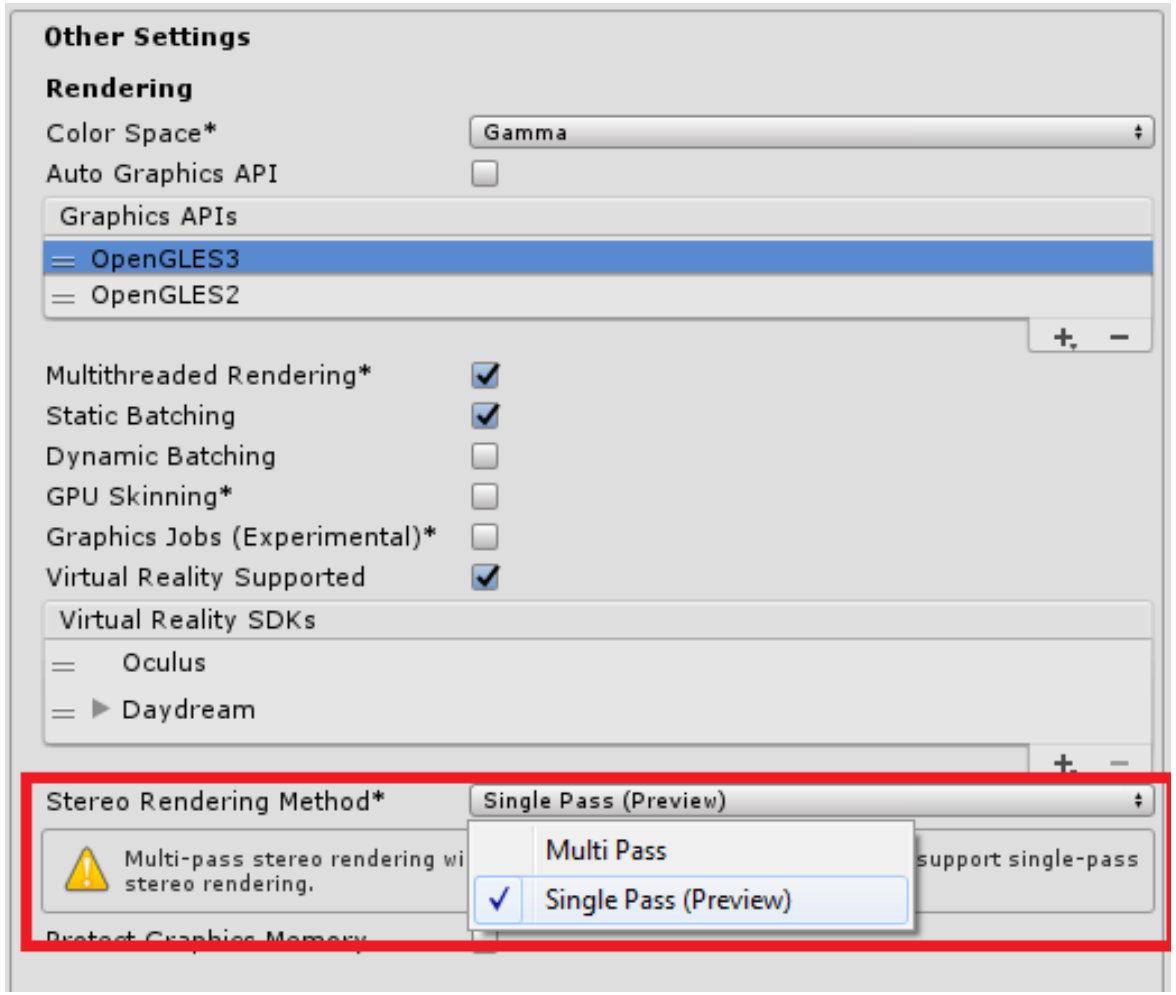


unity_StereoEyeIndex

Single-pass stereo rendering: stereo reflections



Enable single-pass stereo rendering!



ARM

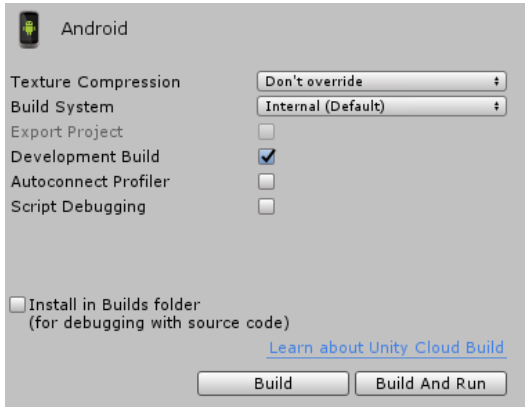
Mali Graphics Debugger (MGD) integration in Unity

Mali Graphics Debugger (MGD)

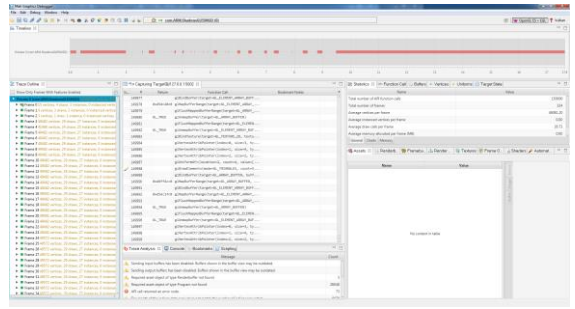
- **Draw-call by Draw-call stepping**
  To identify draw call related issues, redundant draw calls and other opportunities to optimize
- **Texture View**
  Visualize an application's texture usage, to identify opportunities to compress textures or change formats.
- **Shader Statistics**
  Understand which vertex and fragment shaders are the most expensive with cycle count reporting
- **Vertex Attribute / Uniform View**
  See vertex data and index buffers
- **State View**
  Full visibility of graphics state and tracking of state changes
- **Dynamic Optimization Advice**
  Highlighting of common API misusage and dynamic performance improvement advice

Mali Graphics Debugger

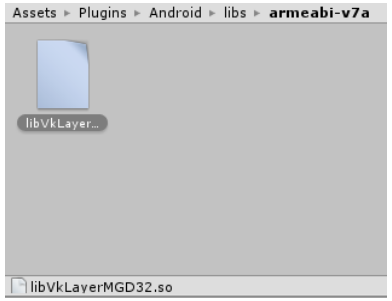
OpenGL ES

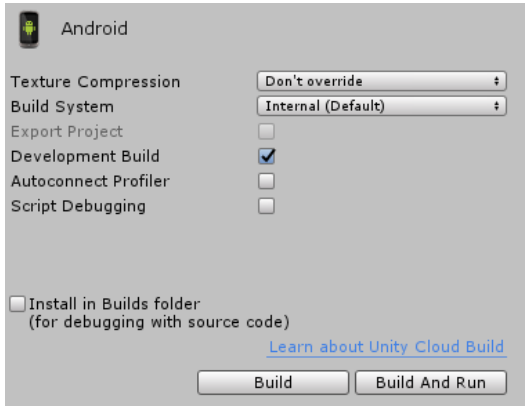



```
adb install -r MGD.apk
adb forward tcp:5002 tcp:5002
```

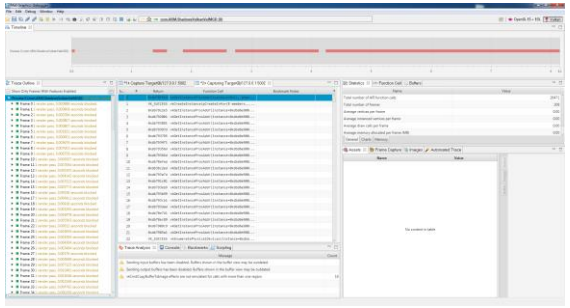


Vulkan






```
adb install -r MGD.apk
adb forward tcp:5002 tcp:5002

adb shell
setprop debug.vulkan.layers
VK_LAYER_ARM_MGD
```

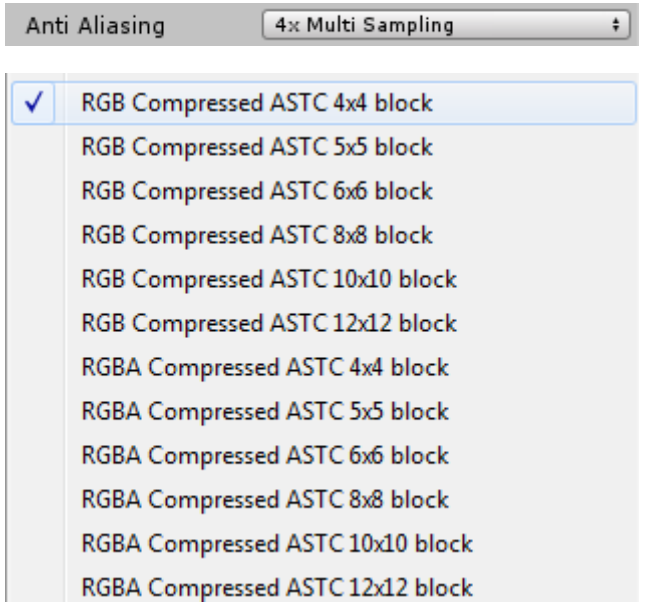


Mobile VR best practice

Mobile VR best practice

Enable 4 x Multisampling Anti Aliasing

- Virtually for free in ARM Mali GPUs



Use texture compression

- ASTC provides wide range of choices

Use optimized rendering techniques

- Allow better performance with less use of resources

Optimized rendering techniques based on local cubemaps

Rendering techniques based on local cubemaps

Technique	Cubemap	Local Correction to
Dynamic soft shadows *	Renders the transparency of scene's boundaries to alpha channel	Vector from fragment to light
Reflection **	Renders scene to RGB channels	Reflection vector
Refraction ***	Renders scene to RGB channels	Refraction vector

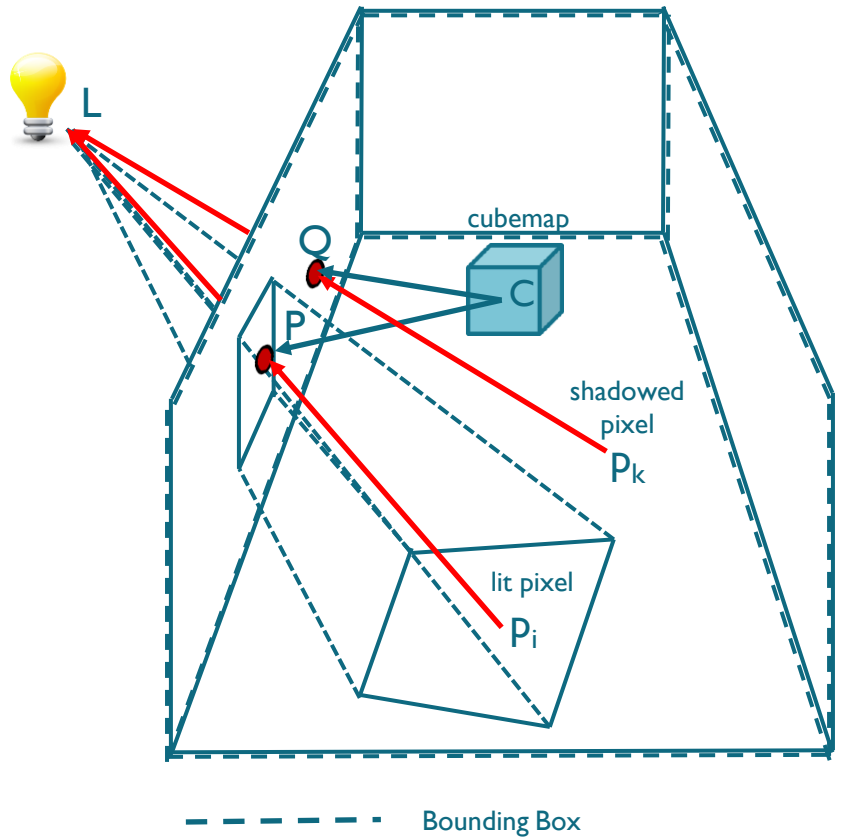
* Unity Asset Store: https://www.assetstore.unity3d.com/en/#!/content/61640

** https://community.arm.com/groups/arm-mali-graphics/blog/2016/03/10/combined-reflections-stereo-reflections-in-vr

*** http://community.arm.com/groups/arm-mali-graphics/blog/2015/04/13/refraction-based-on-local-cubemaps

Dynamic soft shadows based on local cubemaps

Runtime stage



- Create a vertex to light source L vector in the vertex shader.
- Pass this vector to the fragment shader to obtain the vector from the pixel to the light position p_iL.
- Find the intersection of the vector p_iL with the bounding box.
- Build the vector CP from the cubemap position C to the intersection point P.
- Use the new vector CP to fetch the texture from the cubemap.

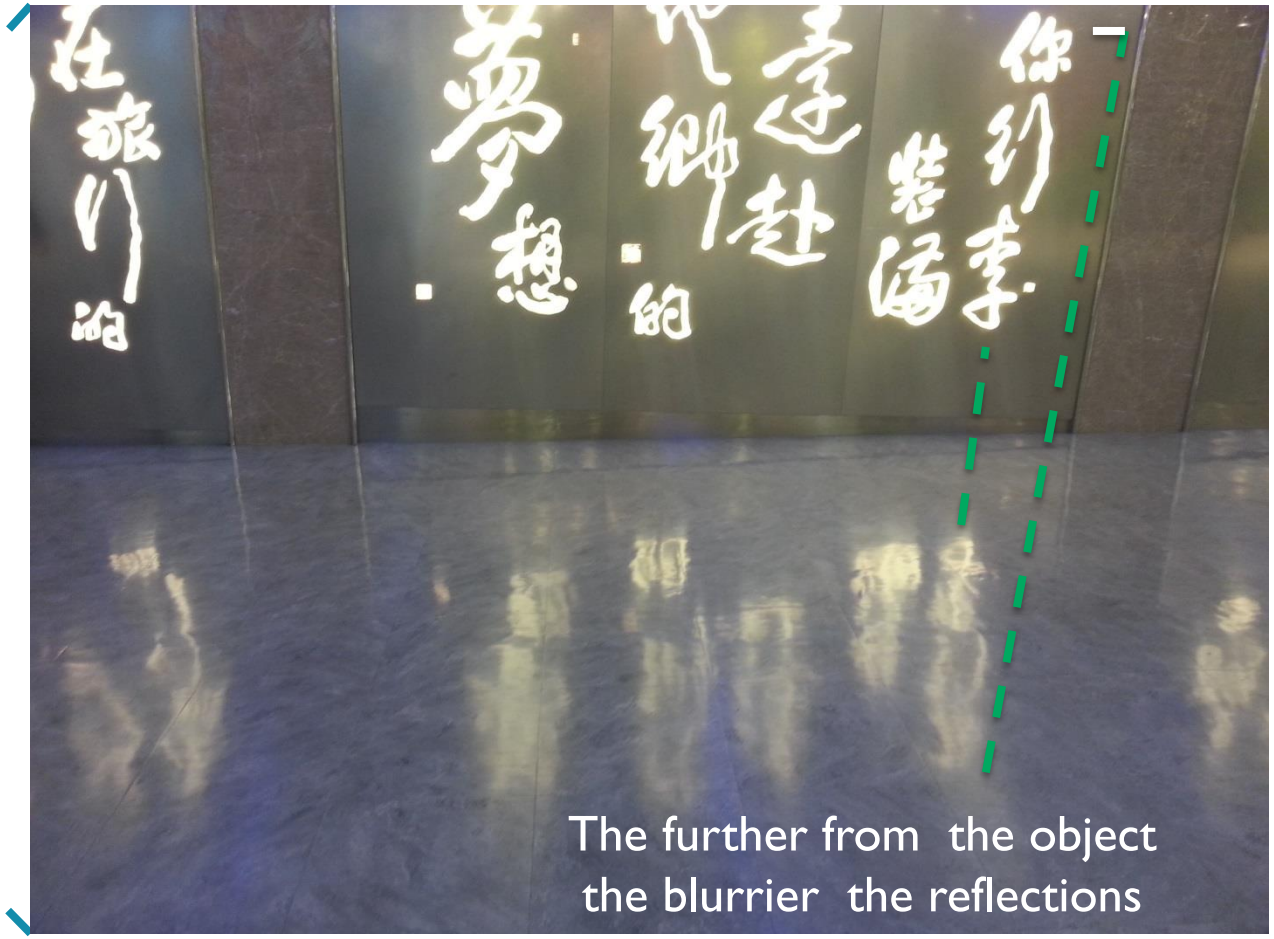
```
float texShadow = texCUBE(_CubeShadows,  CP).a;
```

Source code in the Unity Asset Store

Dynamic soft shadows based on local cubemaps

Reflections based on local cubemap

Blurred Reflections at the Taiwan Intern. Airport



ARM

Blurred reflections based on local cubemaps

Why use rendering techniques based on LC?

Advantages over runtime rendering techniques

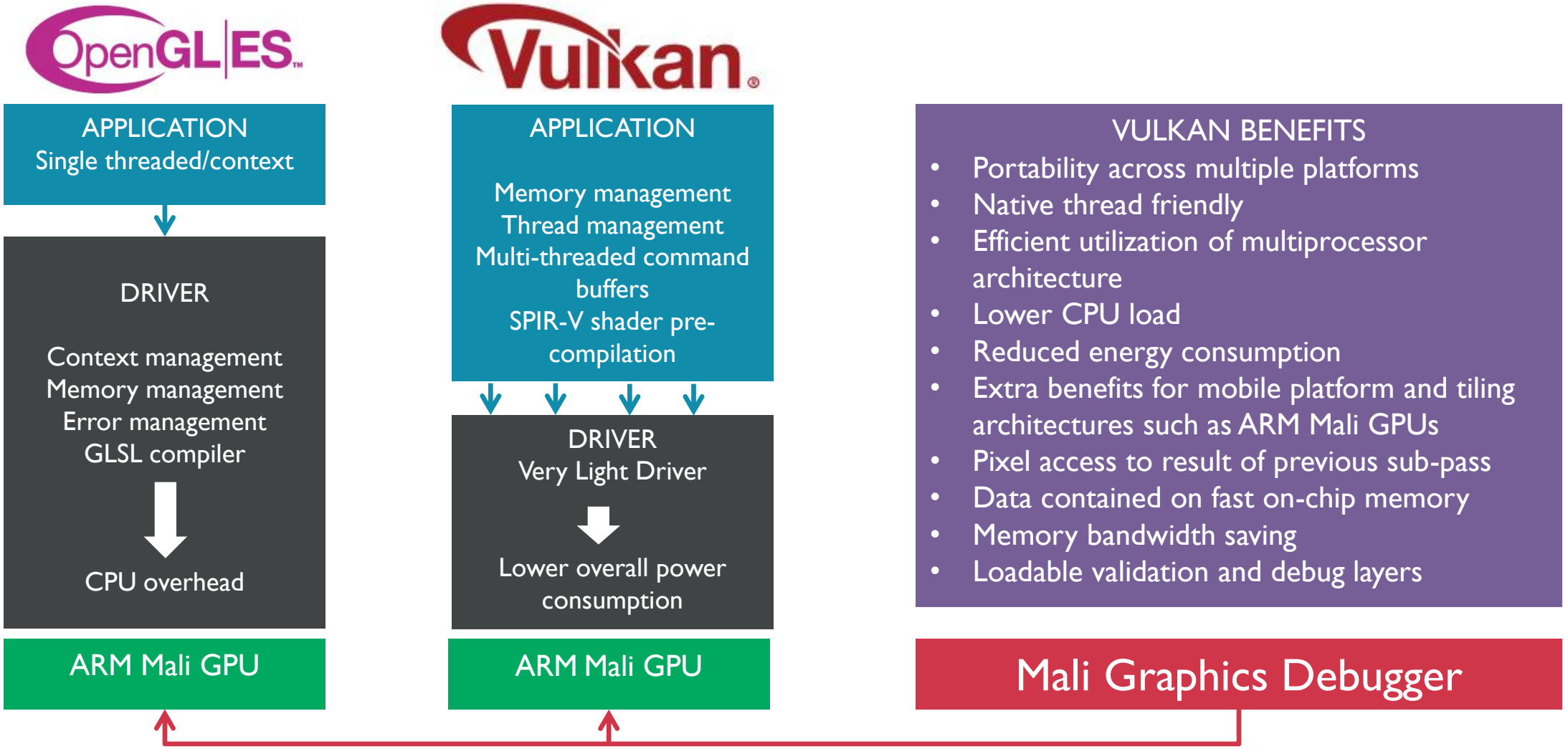
1. Up to 1.5 – 2.8 times faster
2. Resource saving. Bandwidth halved as only read operations
3. Higher quality. No pixel flickering
4. Allow implementing nice effects: soft shadow, blurred reflections and refractions

When possible use rendering techniques based on LC

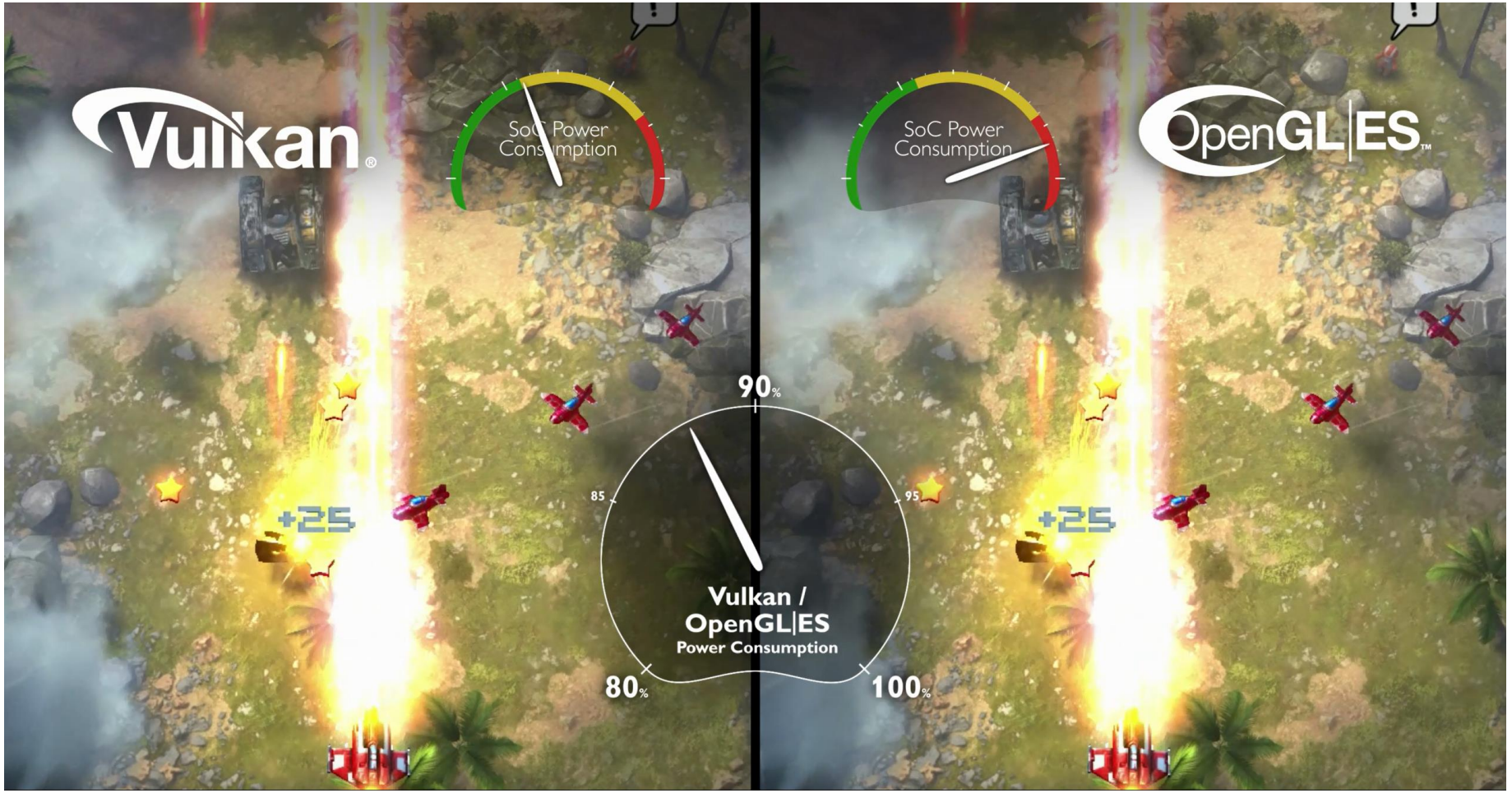
When combined with runtime rendering it helps improving quality at low cost

What next in VR?

Vulkan benefits also expected in VR



Vulkan vs OpenGL ES



Click on the image to watch the video.

Wrap up

- Update to Unity 5.6 to benefit from:
  - Native support for Daydream and Google Cardboard
  - MGD integration in Unity - easier and faster
  - Single pass stereo rendering - less CPU and vertex processing load

- Some recommendations to improve VR performance and quality
  - 4x MSAA - virtually for free in ARM Mali GPUs
  - ASTC - wide range of compression ratios, support for 3D textures
  - Shadows, refraction and reflections based on local cubemaps - faster, resource saving and better quality

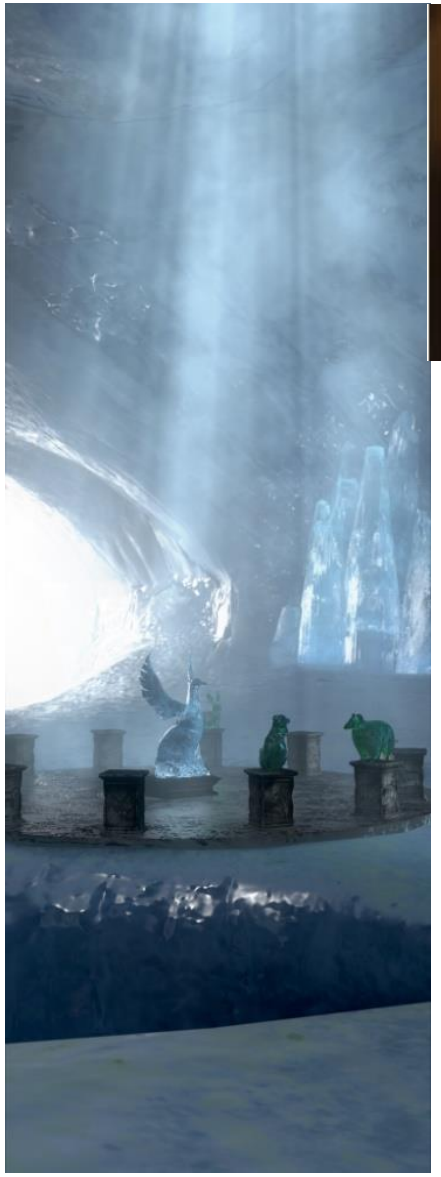
- Vulkan coming to mobile VR
  - Performance and energy consumption improvement expected

Thank you

ARM

Questions

To find out more….



- Find out more info about the topics of this talk at:
  - https://www.youtube.com/watch?v=mb98QOlZ8ZE
  - https://community.arm.com/graphics/b/blog/posts/optimizing-virtual-reality-understanding-multiview
  - https://community.arm.com/graphics/b/blog/posts/mgd-integration-in-unity
  - https://community.arm.com/graphics/b/blog/posts/intro-to-astc-as-presented-at-cgdc-2013
  - http://malideveloper.arm.com/armunityguide
  - https://community.arm.com/groups/arm-mali-graphics/blog/2016/03/10/combined-reflections-stereo-reflections-in-vr
  - https://community.arm.com/groups/arm-mali-graphics/blog/2016/04/20/achieving-high-quality-mobile-vr-games
  - http://community.arm.com/groups/arm-mali-graphics/blog/2015/04/13/dynamic-soft-shadows-based-on-local-cubemap
  - http://community.arm.com/groups/arm-mali-graphics/blog/2014/08/07/reflections-based-on-local-cubemaps
  - http://community.arm.com/groups/arm-mali-graphics/blog/2015/04/13/refraction-based-on-local-cubemaps
  - http://community.arm.com/groups/arm-mali-graphics/blog/2015/05/21/the-power-of-local-cubemaps-at-unite-apac-and-the-taoyuan-effect
  - https://www.youtube.com/watch?v=WI7nXq8oozw
  - https://www.assetstore.unity3d.com/en/#!/content/61640

ARM