High quality mobile VR with Unreal Engine and Oculus

Daniele Di Donato

Senior Software Engineer, ARM

Rémi Palandri

Graphics Engineer, Oculus

Ryan Vance

Senior Rendering Programmer, Epic Games

GDC’17

3/1/2017

© ARM 2017

Agenda

- VR Best practises on GearVR / UE4
- New VR features and rendering techniques in UE4
  - Monoscopic Far Field Rendering
  - Mobile Multiview
- New technologies in the horizon using Multiview
  - Foveated Rendering
- Debugging and Profiling on Mali
  - Mali Offline Shader Compiler
  - Mali Graphics Debugger
  - Streamline

© ARM 2017

Best practices

- Compared to PC and Console, Mobile has additional constraints
  - Most accessible development environment
    - Most challenging platform to ship on
  - Battery life and heat dissipation are primary concerns
    - Fast peak performance, but you can't run it pegged indefinitely
  - Optimization is more involved than PC and Console
    - Consistently making frame rate isn't enough
  - Android N sustained performance mode
    - Guaranteed to run indefinitely at a lower performance level

© ARM 2017

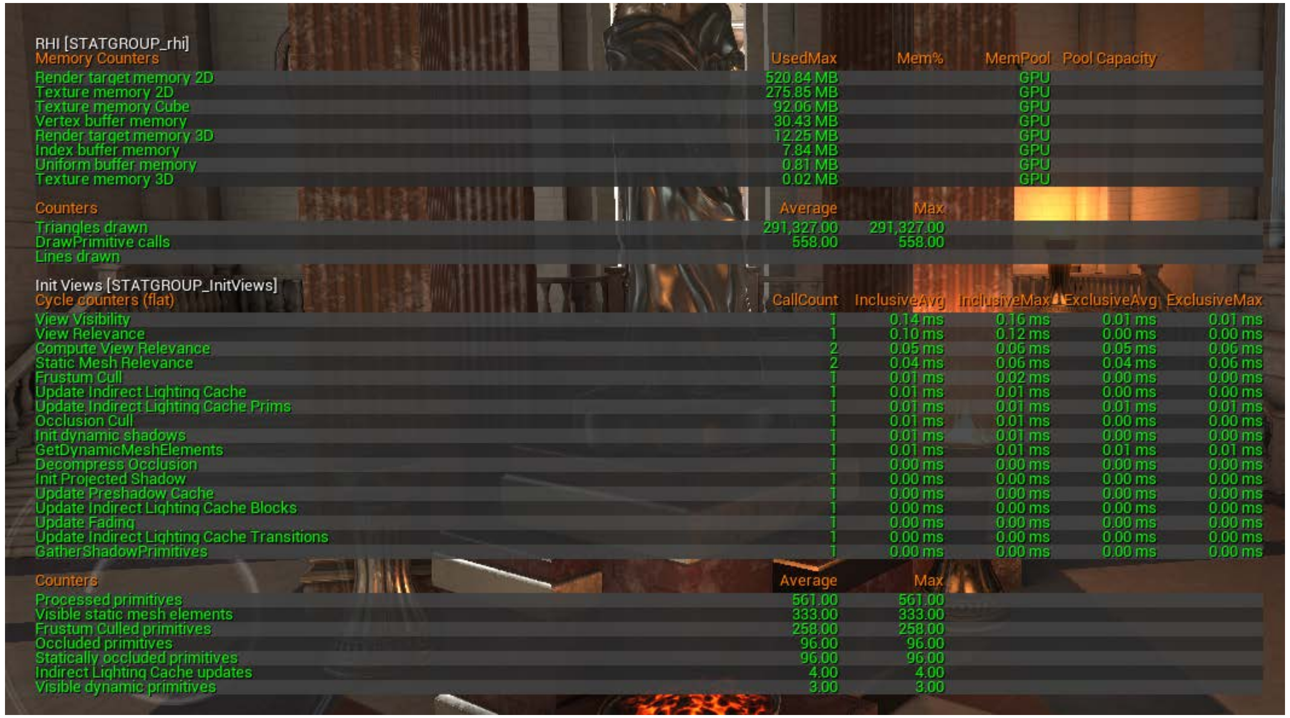
Best practices

- Asset budgets and suggestions
  - 50 - 60k average number of triangles for the entire scene
    - Max 100k
  - 50 draw calls per eye
    - Merge materials and meshes
    - Use instancing
    - Multiview improves this!
  - Aggressive LODs
    - Consider the memory impact

© ARM 2017

Best practices

- Use the stat system to profile scene complexity



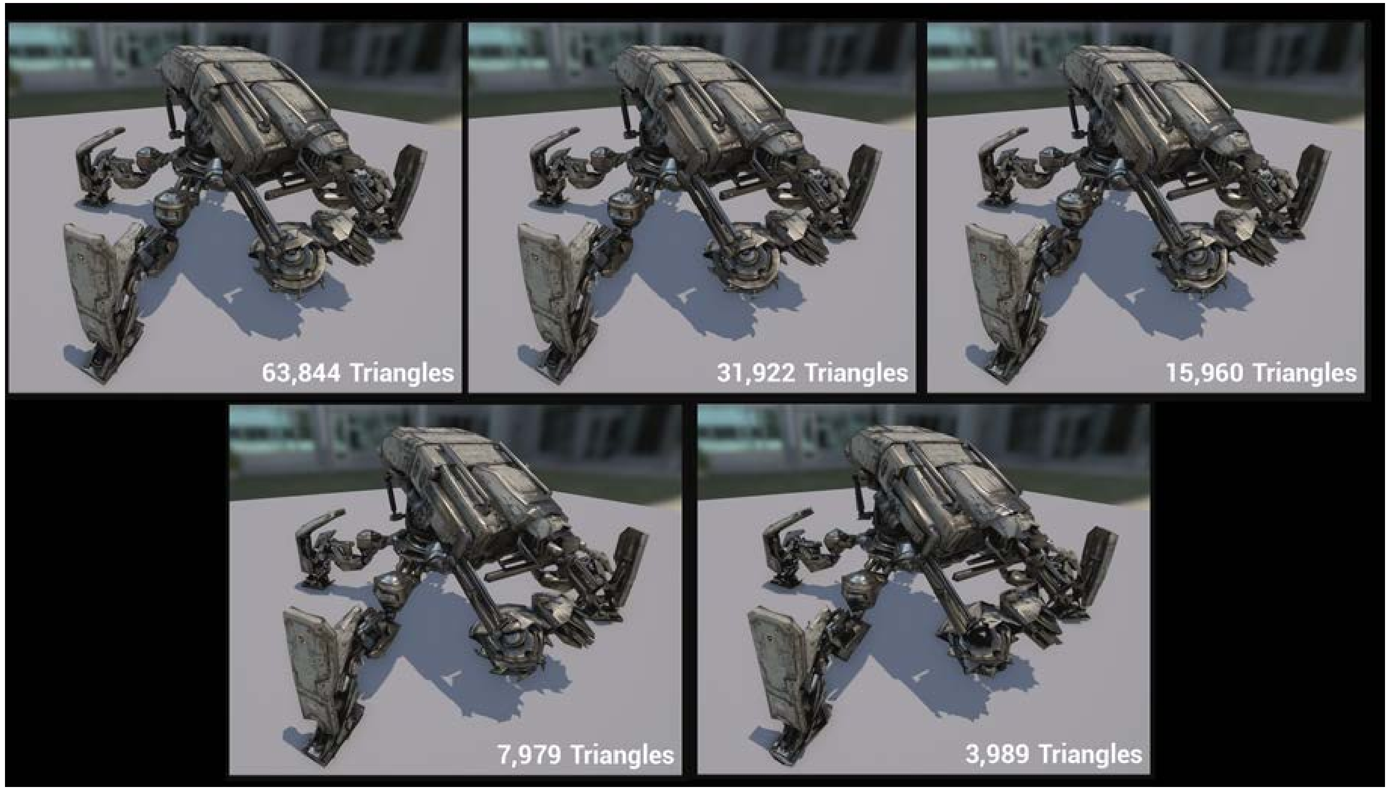
```
RHI [STATGROUP_rhi]
Memory Counters                               UsedMax      Mem%    MemPool  Pool Capacity
Render target memory 2D                     520.84 MB                  GPU
Texture memory 2D                           275.85 MB                  GPU
Texture memory Cube                          92.06 MB                  GPU
Vertex buffer memory                         30.43 MB                  GPU
Render target memory 3D                      12.25 MB                  GPU
Index buffer memory                           7.84 MB                  GPU
Uniform buffer memory                         0.81 MB                  GPU
Texture memory 3D                             0.02 MB                  GPU

Counters                                      Average        Max
Triangles drawn                            291,327.00 291,327.00
DrawPrimitive calls                            558.00     558.00
Lines drawn

Init Views [STATGROUP_InitViews]
Cycle counters (flat)                       CallCount InclusiveAvg InclusiveMax ExclusiveAvg ExclusiveMax
View Visibility                                     1      0.14 ms      0.16 ms      0.01 ms      0.01 ms
View Relevance                                      1      0.10 ms      0.12 ms      0.00 ms      0.00 ms
Compute View Relevance                              2      0.05 ms      0.06 ms      0.05 ms      0.06 ms
Static Mesh Relevance                               2      0.04 ms      0.06 ms      0.04 ms      0.06 ms
Frustum Cull                                        1      0.01 ms      0.02 ms      0.00 ms      0.00 ms
Update Indirect Lighting Cache                      1      0.01 ms      0.01 ms      0.00 ms      0.00 ms
Update Indirect Lighting Cache Prims                1      0.01 ms      0.01 ms      0.01 ms      0.01 ms
Occlusion Cull                                      1      0.01 ms      0.01 ms      0.00 ms      0.00 ms
Init dynamic shadows                                1      0.01 ms      0.01 ms      0.00 ms      0.00 ms
GetDynamicMeshElements                              1      0.01 ms      0.01 ms      0.01 ms      0.01 ms
Decompress Occlusion                                1      0.00 ms      0.00 ms      0.00 ms      0.00 ms
Init Projected Shadow                               1      0.00 ms      0.00 ms      0.00 ms      0.00 ms
Update Preshadow Cache                              1      0.00 ms      0.00 ms      0.00 ms      0.00 ms
Update Indirect Lighting Cache Blocks               1      0.00 ms      0.00 ms      0.00 ms      0.00 ms
Update Fading                                       1      0.00 ms      0.00 ms      0.00 ms      0.00 ms
Update Indirect Lighting Cache Transitions          1      0.00 ms      0.00 ms      0.00 ms      0.00 ms
GatherShadowPrimitives                              1      0.00 ms      0.00 ms      0.00 ms      0.00 ms

Counters                                      Average        Max
Processed primitives                           561.00     561.00
Visible static mesh elements                   333.00     333.00
Frustum Culled primitives                      258.00     258.00
Occluded primitives                             96.00      96.00
Statically occluded primitives                  96.00      96.00
Indirect Lighting Cache updates                  4.00       4.00
Visible dynamic primitives                       3.00       3.00
```

Output of stat rhi and initviews

© ARM 20 17

Best practices

- Example output of our new automatic LOD generation (4.14)
  - Vertex data is maintained so materials and light maps can be shared



© ARM 2017

Best practices

- Asset budgets and suggestions
  - Materials should be no more than 125 instructions
  - No dynamic lights or shadows
    - Bake and fake
  - LDR
    - No post processing
  - Create test levels with representative content
    - Profile on devices you intend to ship on to verify your budgets
    - Test for the duration of expected session time

Best practices

- Content suggestions
  - Remove triangles the user can’t see
    - Remove back sides
    - Segment large models
  - Bake distant environment to a skybox
    - Use Oculus cube map layer for optimal sampling
    - Monoscopic can be used for middle ground
  - Fully rough materials
    - Fake environment reflection

© ARM 2017

Best practices

- Content suggestions
  - Don’t render occluded objects
    - Wasted draw calls and primitive culling time
  - Design scenes to minimize draw distance
  - Use precomputed visibility volumes
  - Aggressive manual hiding of objects not in view
    - Take advantage of scene knowledge
  - Minimize transparent overdraw
  - Objects that are 100% transparent are still drawn. Set visibility flag!

© ARM 2017

Best practices

- Content suggestions
  - Use MSAA
    - At least 2x, and 4x when possible
    - Avoid post process anti-aliasing
  - Use ASTC for texture compression
    - Largest block size possible
    - Generate MIP maps
    - Avoid complex filtering options

© ARM 2017

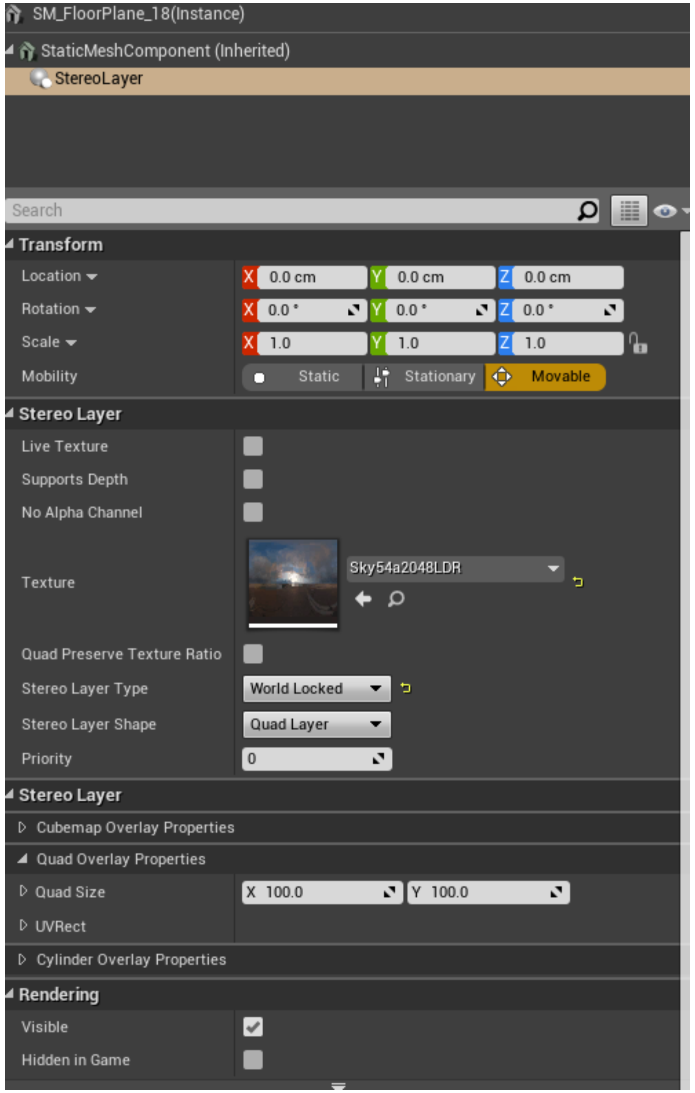
Best practices

- Content suggestions
  - Track ticking object count
    - Don't tick if you don't need to
  - Spawning is extremely expensive
    - Spawn on load
    - Amortize over multiple frames
    - Consider building a manager to pool objects
  - Try blueprint nativization to reduce script VM overhead

© ARM 2017

Best practices

- Stereo Layers
  - Not rendered in-engine
  - Raytraced in the compositor
  - Single sampling!
  - Supports quads, cylinders, and cubemaps
  - Head-locked, tracker-locked, or world-locked
  - Stereo Layer Component
  - Works with UMG!



SM_FloorPlane_18(Instance)

StaticMeshComponent (Inherited)

StereoLayer

Search

Transform

Location	X 0.0 cm	Y 0.0 cm	Z 0.0 cm
Rotation	X 0.0 °	Y 0.0 °	Z 0.0 °
Scale	X 1.0	Y 1.0	Z 1.0
Mobility	Static	Stationary	[✓ Movable]

Stereo Layer

Live Texture	☐
Supports Depth	☐
No Alpha Channel	☐
Texture	Sky54a2048LDR
Quad Preserve Texture Ratio	☐
Stereo Layer Type	World Locked
Stereo Layer Shape	Quad Layer
Priority	0

Stereo Layer

- Cubemap Overlay Properties
- Quad Overlay Properties
  - Quad Size: X 100.0, Y 100.0
  - UVRect
- Cylinder Overlay Properties

Rendering

Visible	☑
Hidden in Game	☐

© ARM 2017

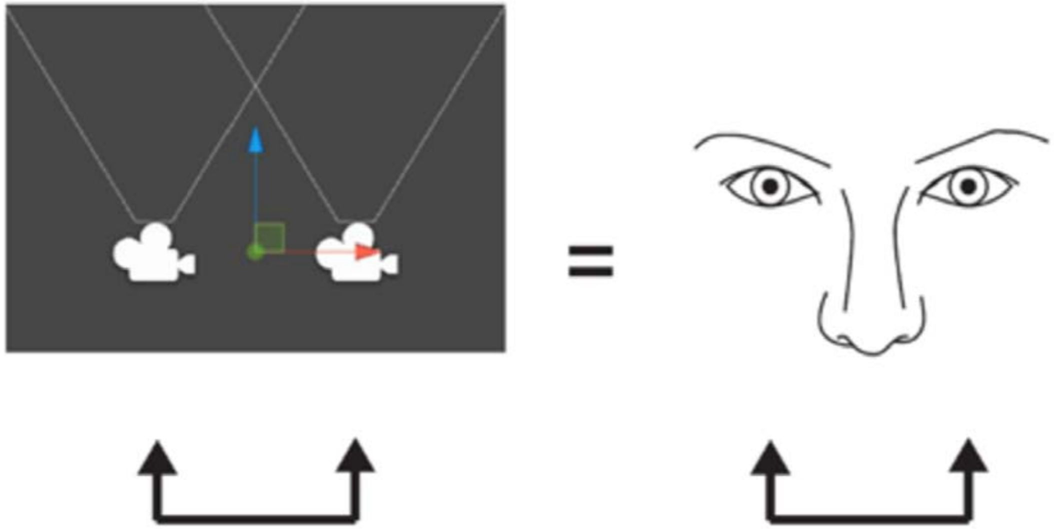
New VR features and rendering techniques in UE4

- Monoscopic Far Field Rendering
- Mobile Multiview

© ARM 2017

Monoscopic Rendering

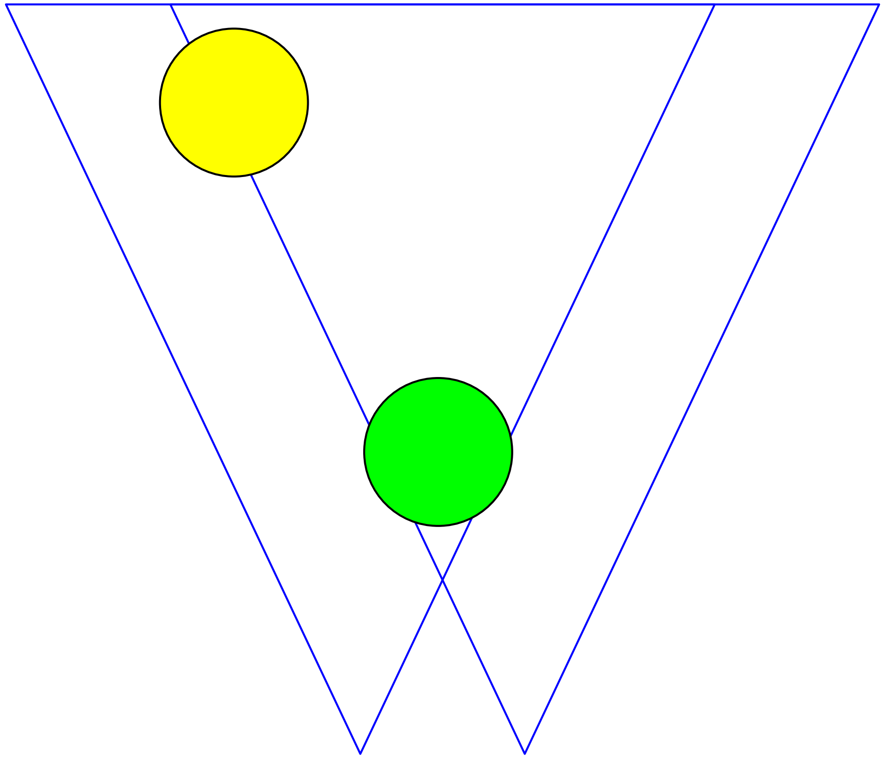
- Rendering both eyes
  - Position difference creates binocular parallax
  - Projection difference creates binocular disparity
  - Depth!
- Performance issues
  - Double the CPU usage
  - Double the Vertex/Fragment usage
- Similarities?



© ARM 2017

Monoscopic Rendering

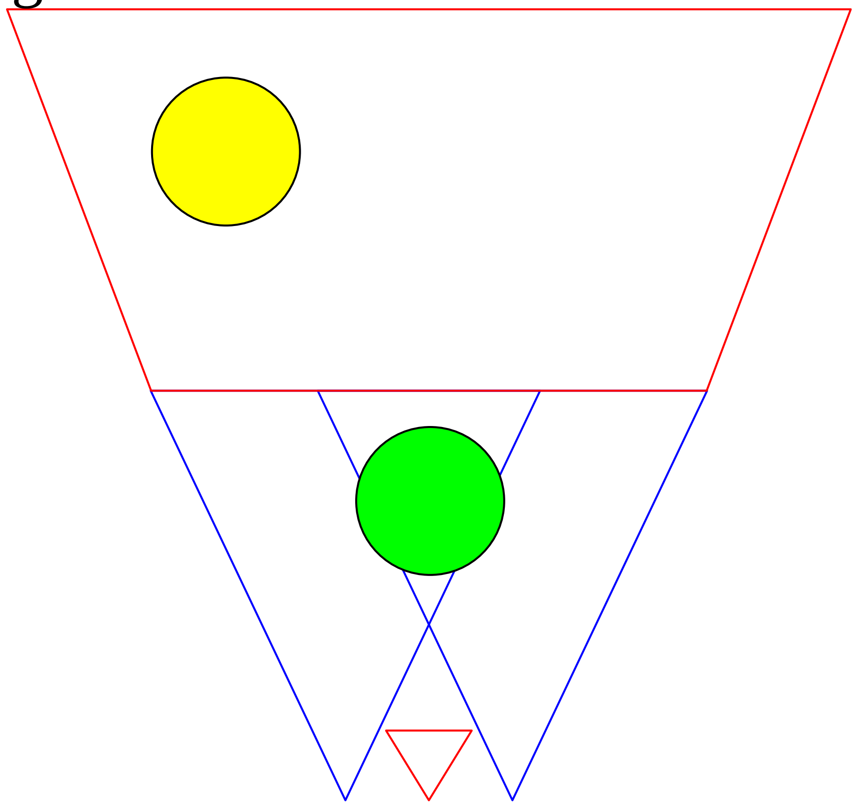
- Position difference less significant as distances grow
- Adding a 3rd camera!
  - Two stereo cameras have a 30 ft far plane
  - Mono camera has a 30 ft near plane
  - Strict ordering of pixels
- New rendering pipeline



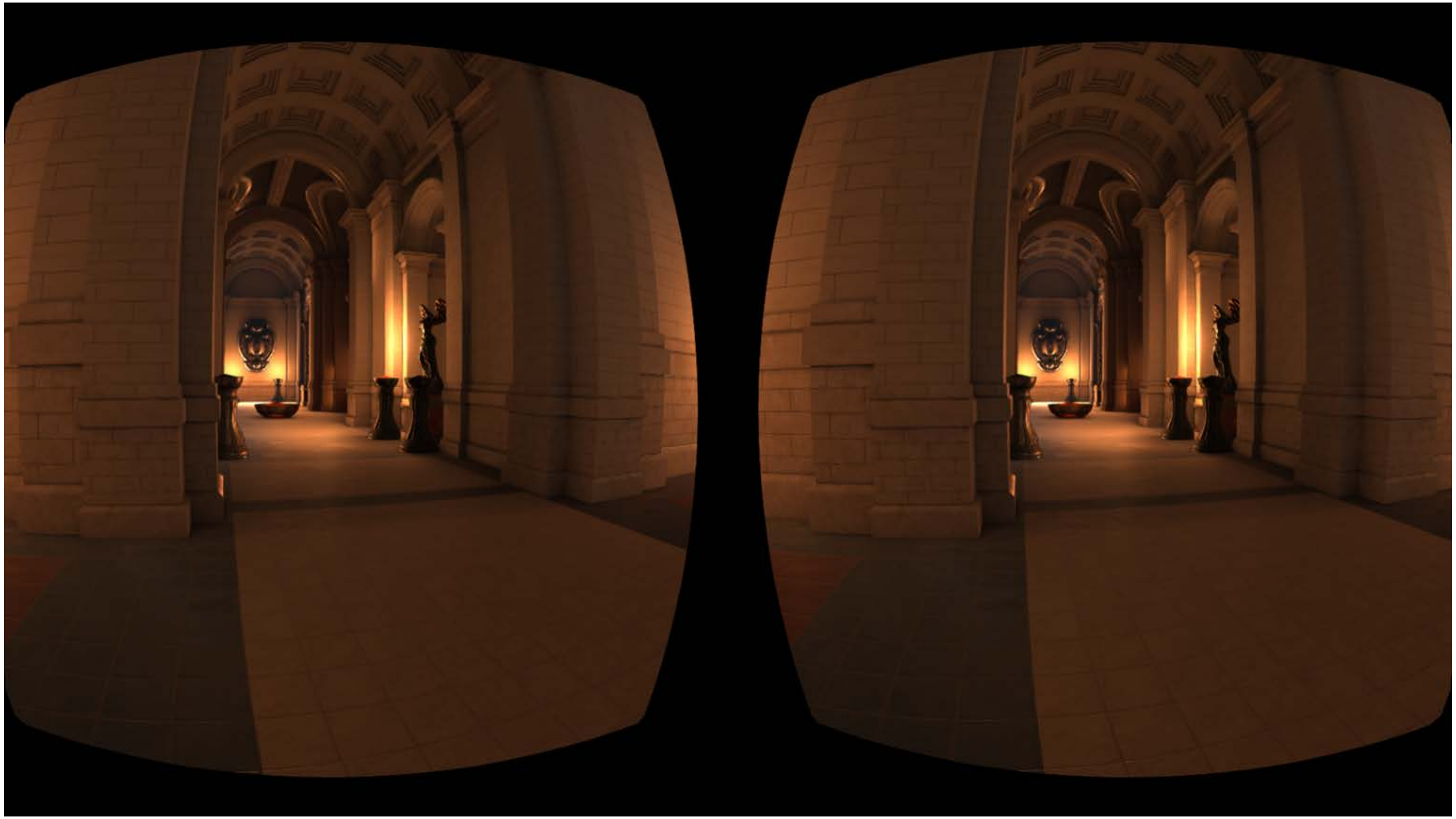
© ARM 2017

Monoscopic Rendering

- Position difference less significant as distances grow
- Adding a 3rd camera!
  - Two stereo cameras have a 30 ft far plane
  - Mono camera has a 30 ft near plane
  - Strict ordering of pixels
- New rendering pipeline

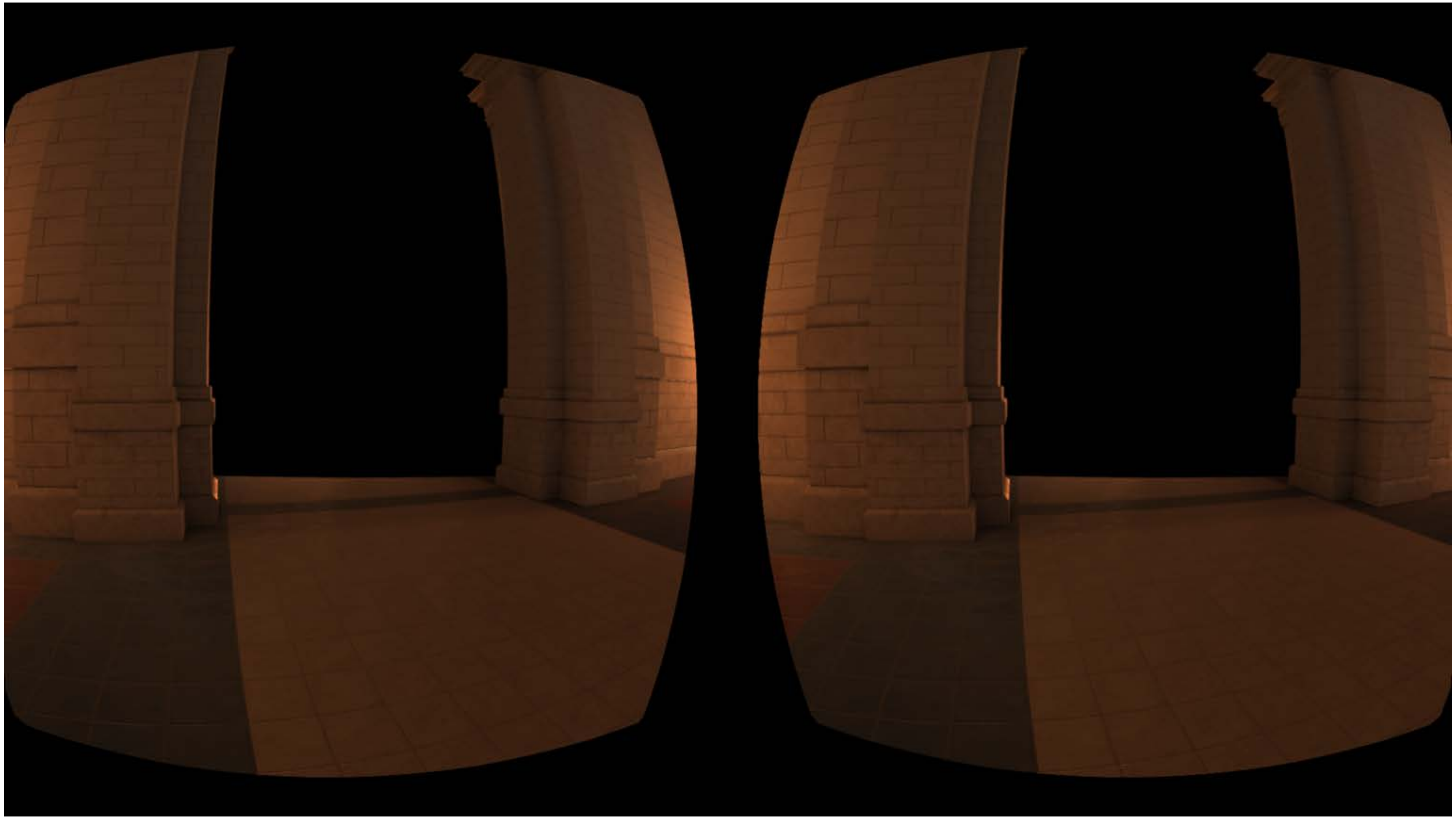


Monoscopic Rendering



Original rendering

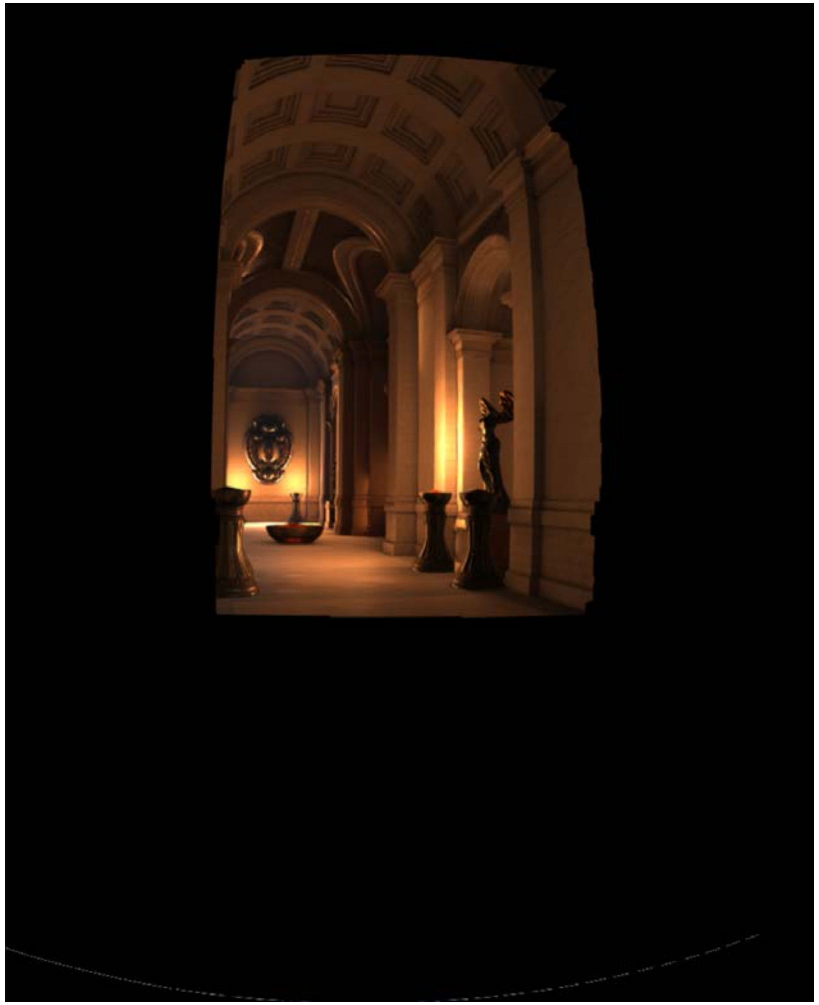
Monoscopic Rendering



Stereo cameras with 30ft far plane

© ARM 2017

Monoscopic Rendering



Mono camera with 30ft near plane

© ARM 2017

Monoscopic Rendering

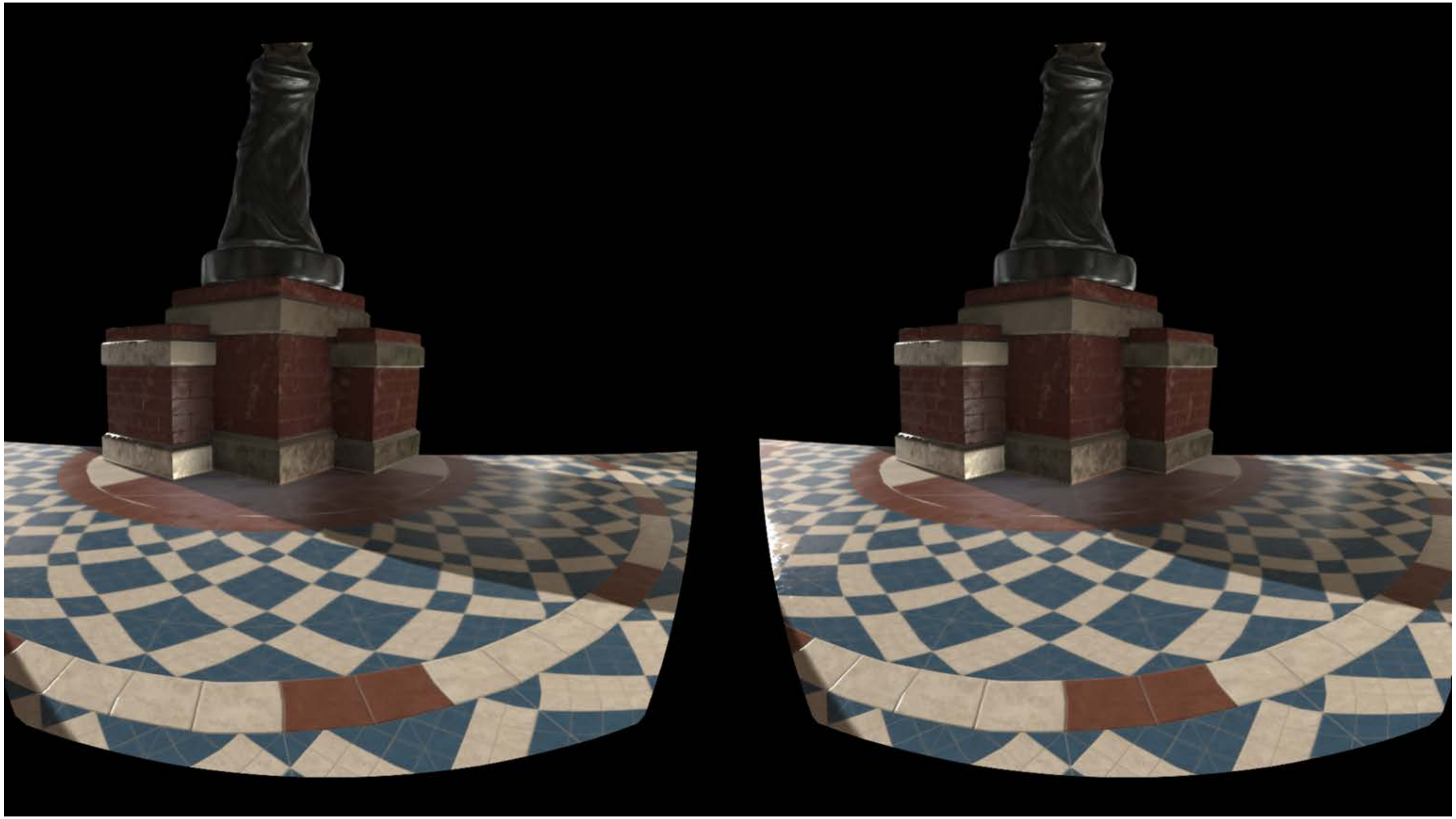
- Issues with new pipeline
  - Monoscopic camera rendering unused pixels

Monoscopic Rendering

Standard SunTemple scene

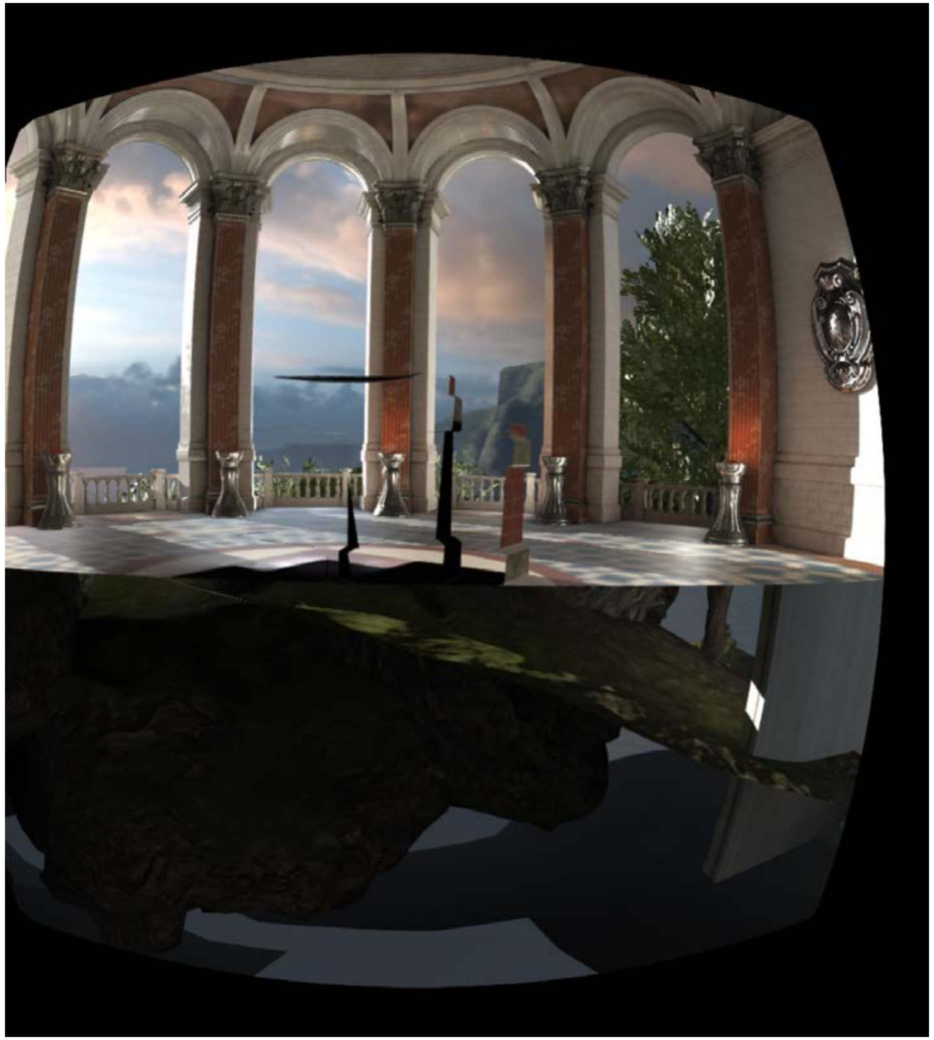
© ARM 20 17

Monoscopic Rendering



Close stereo cameras

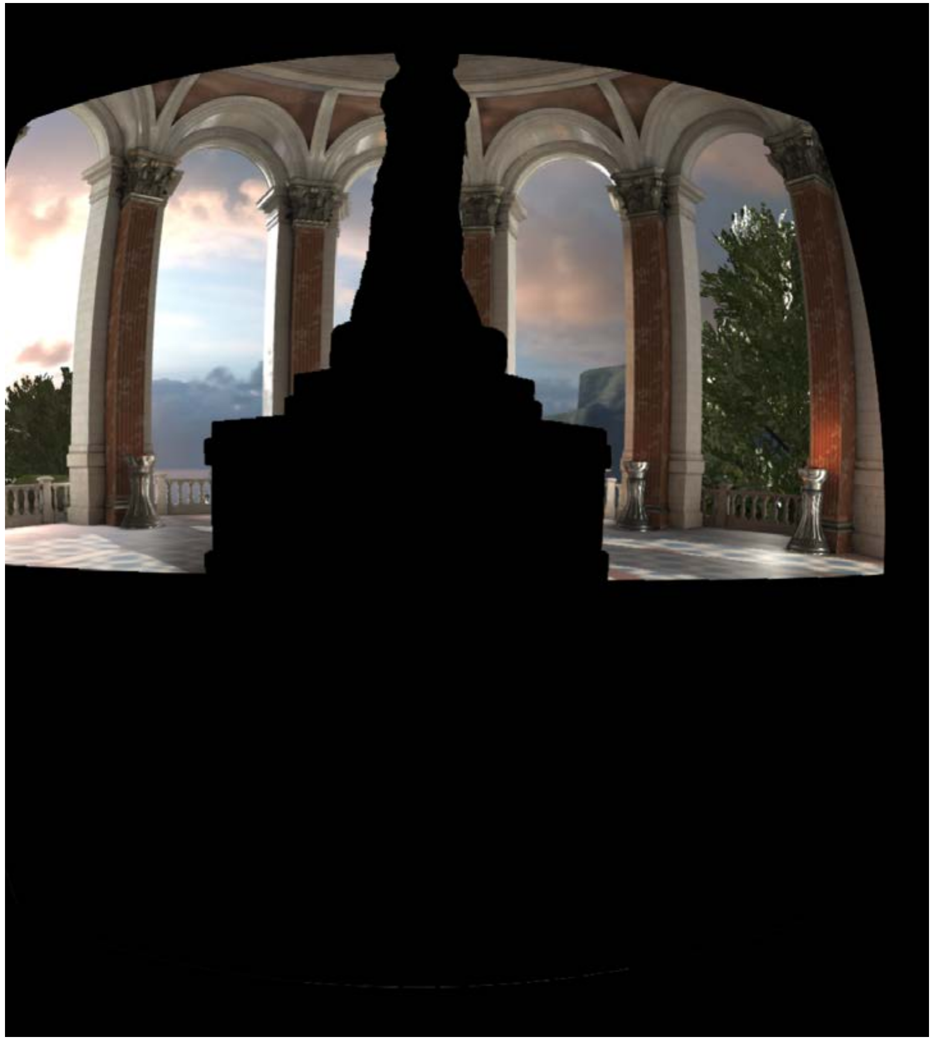
Monoscopic Rendering



Mono camera with 30ft near plane

© ARM 20 17

Monoscopic Rendering



Mono camera with 30ft near plane and mask

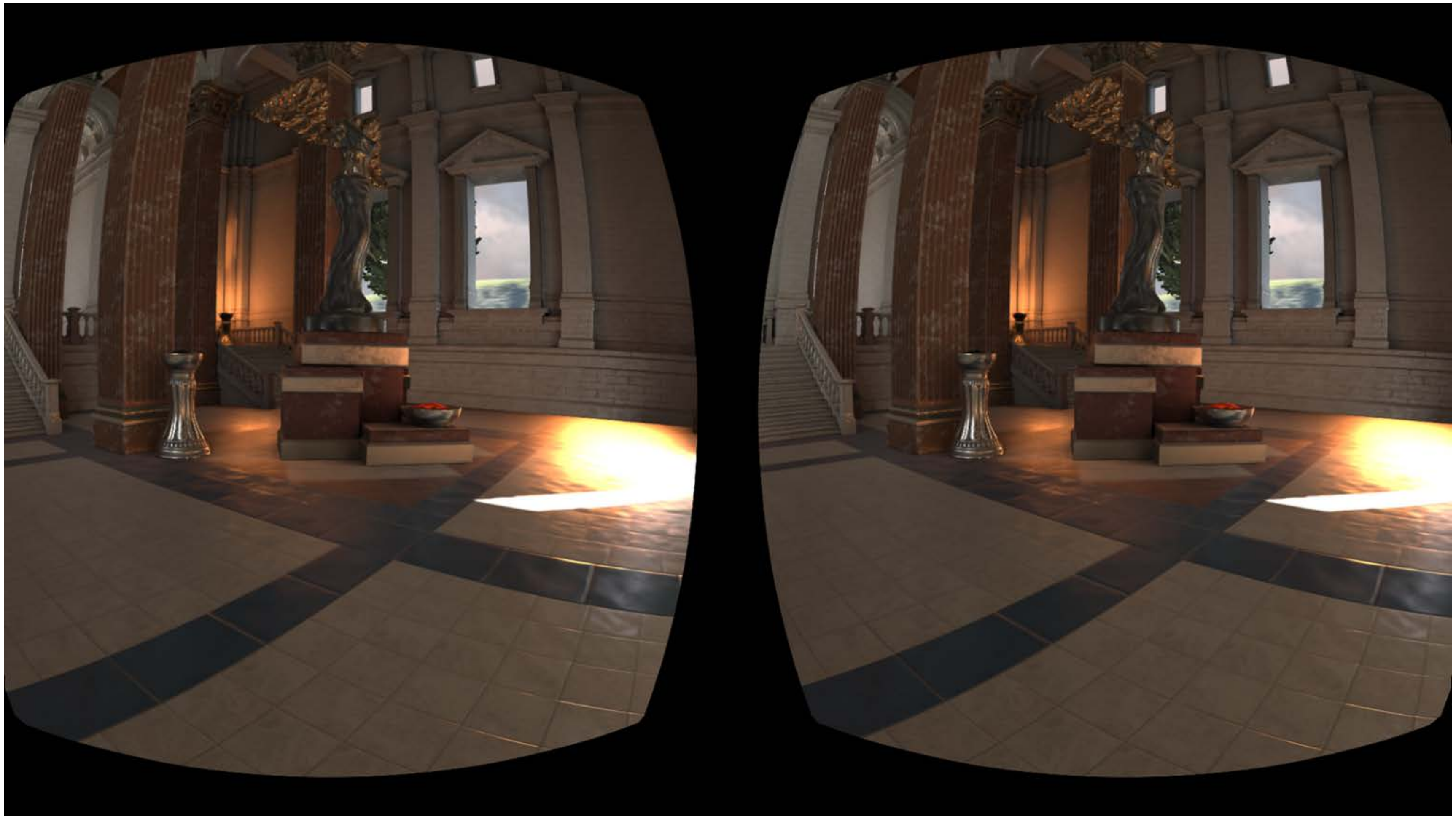
© ARM 2017

Monoscopic Rendering

- Issues with new pipeline
  - Monoscopic camera rendering unused pixels
  - Stereo cameras drawing far object (frustum culling)

© ARM 2017

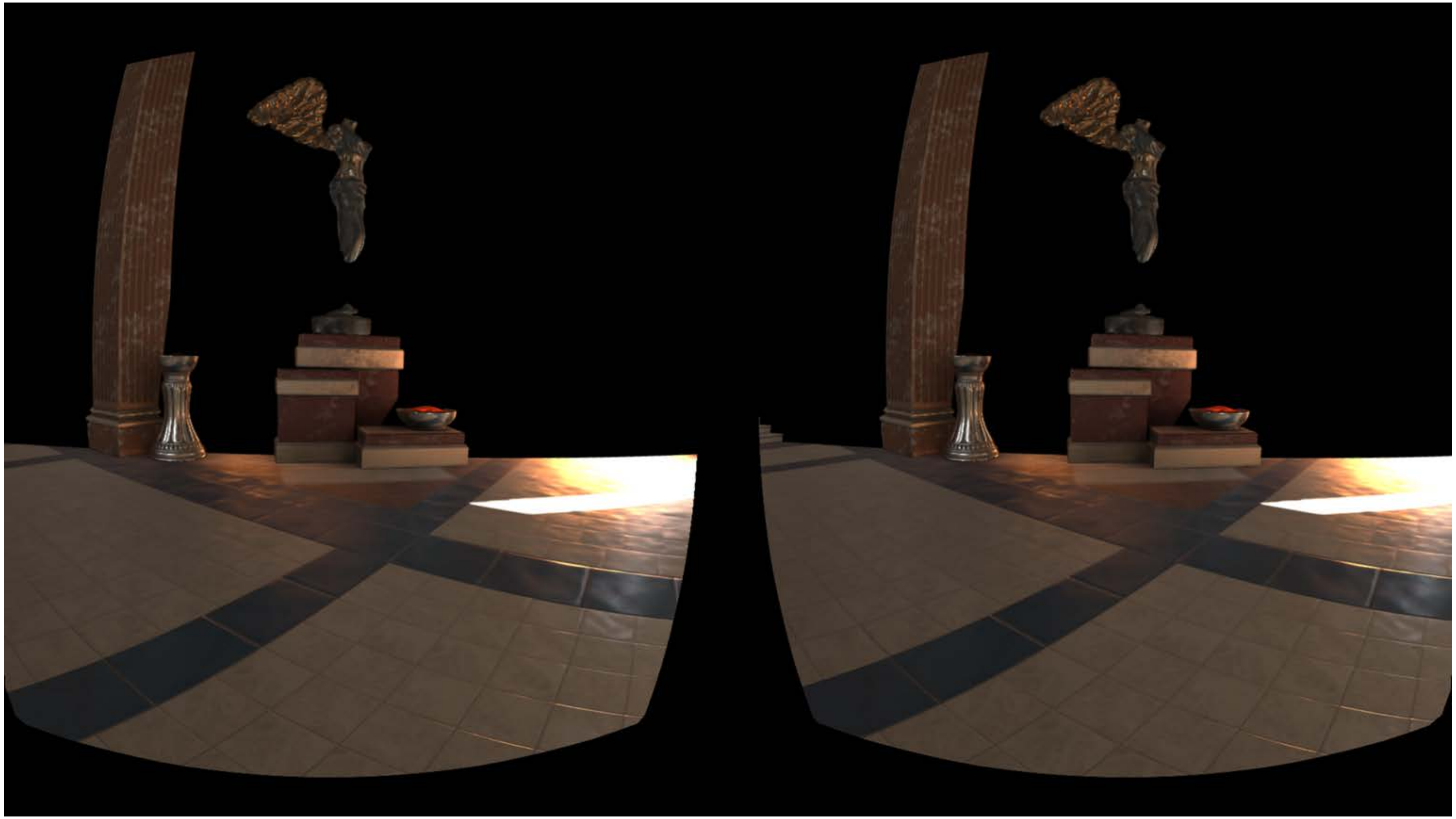
Monoscopic Rendering



Standard SunTemple scene

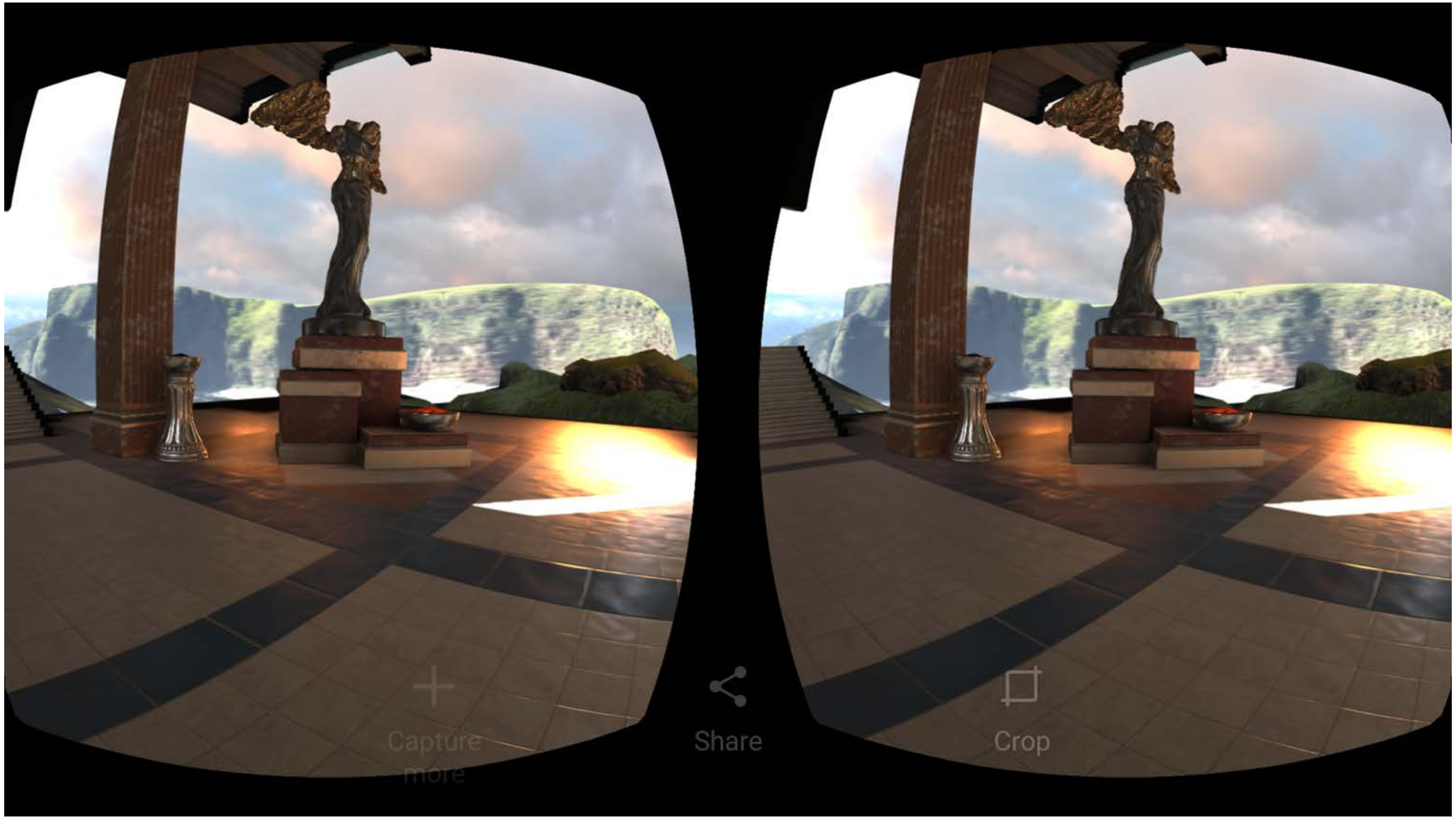
© ARM 2017

Monoscopic Rendering



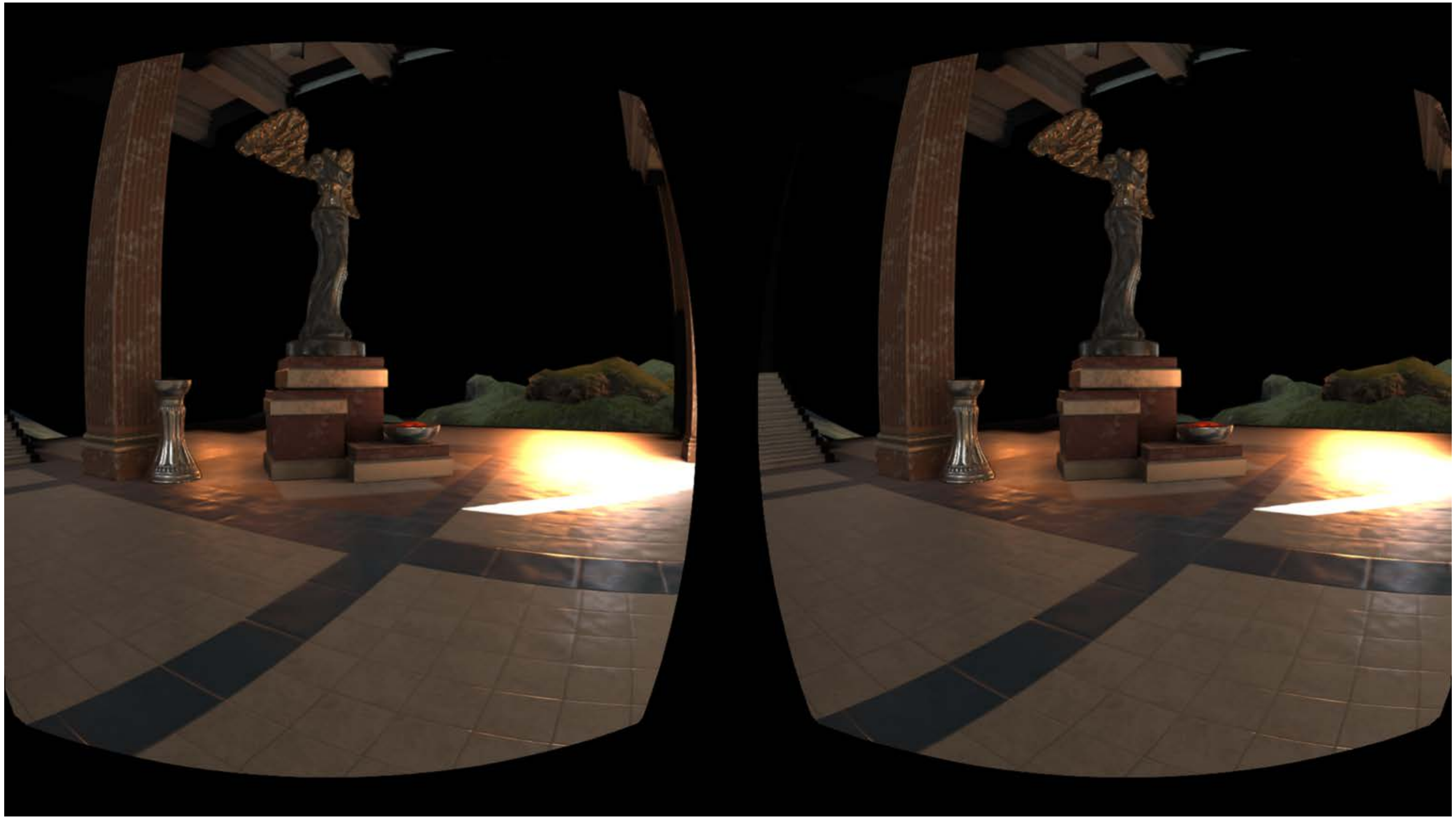
Close stereo cameras

Monoscopic Rendering



Stereo cameras without close depth clear
but with close frustum culling

Monoscopic Rendering



Stereo cameras without close depth clear
but with close frustum culling

© ARM 2017

Monoscopic Rendering

- Issues with new pipeline
  - Monoscopic camera rendering unused pixels
  - Stereo cameras drawing far object (frustum culling)
  - Compositing artifacts (transparency mainly)
  - Performance hits of running a third camera

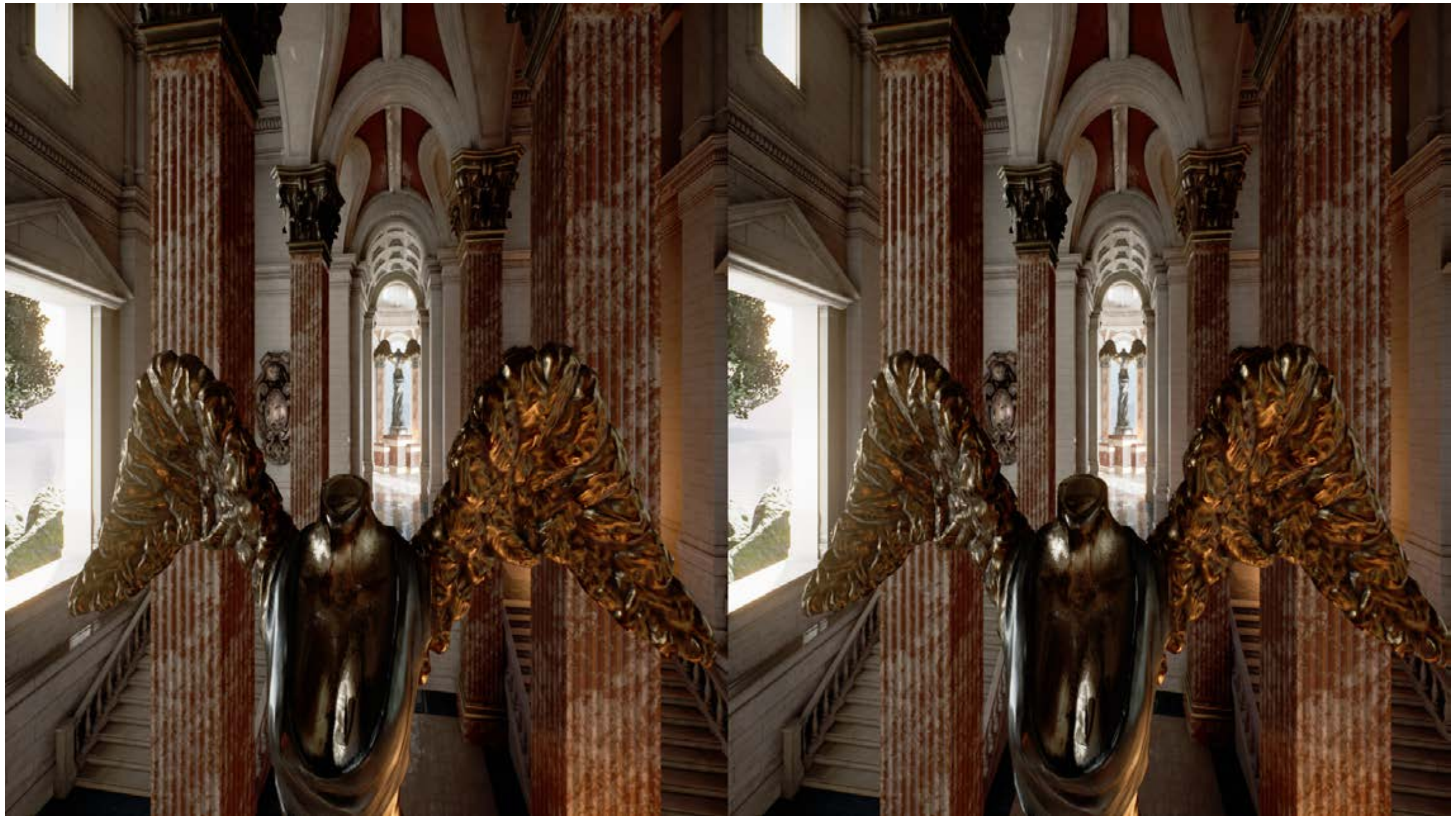
© ARM 2017

Monoscopic Rendering

- Results
  - Performance is very environment-dependent
  - 20 +% increases in certain conditions
  - Both CPU and GPU implications
  - Can get performance decrease as well
  - Dynamic system, both on/off and view distance
  - *vr.FarFieldRenderingMode 0/1/2/3/4*

Mobile Multiview

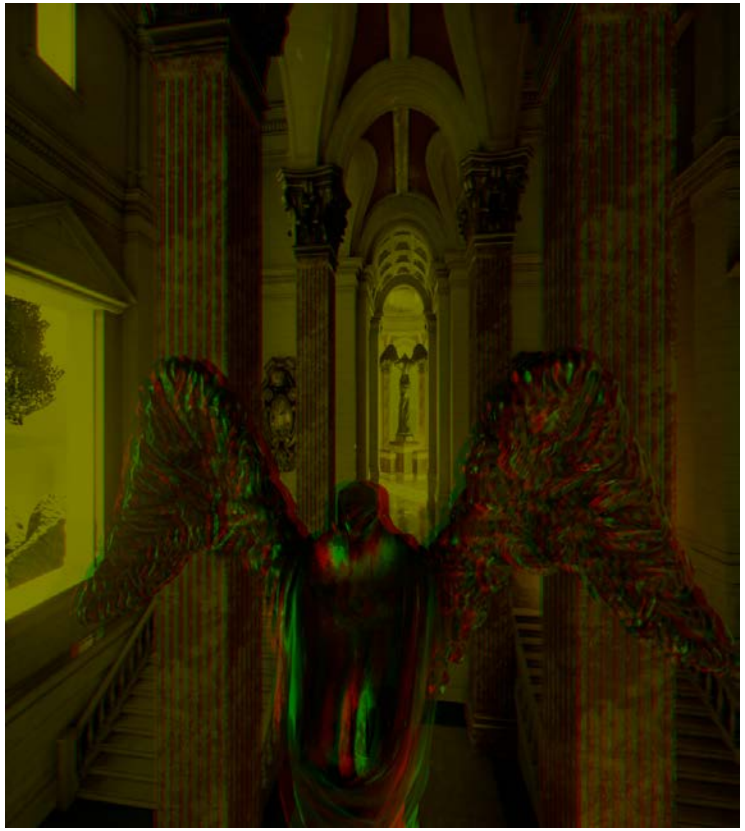
- What problem are we solving?



© ARM 2017

Mobile Multiview

- Minimal differences between views at primitive granularity

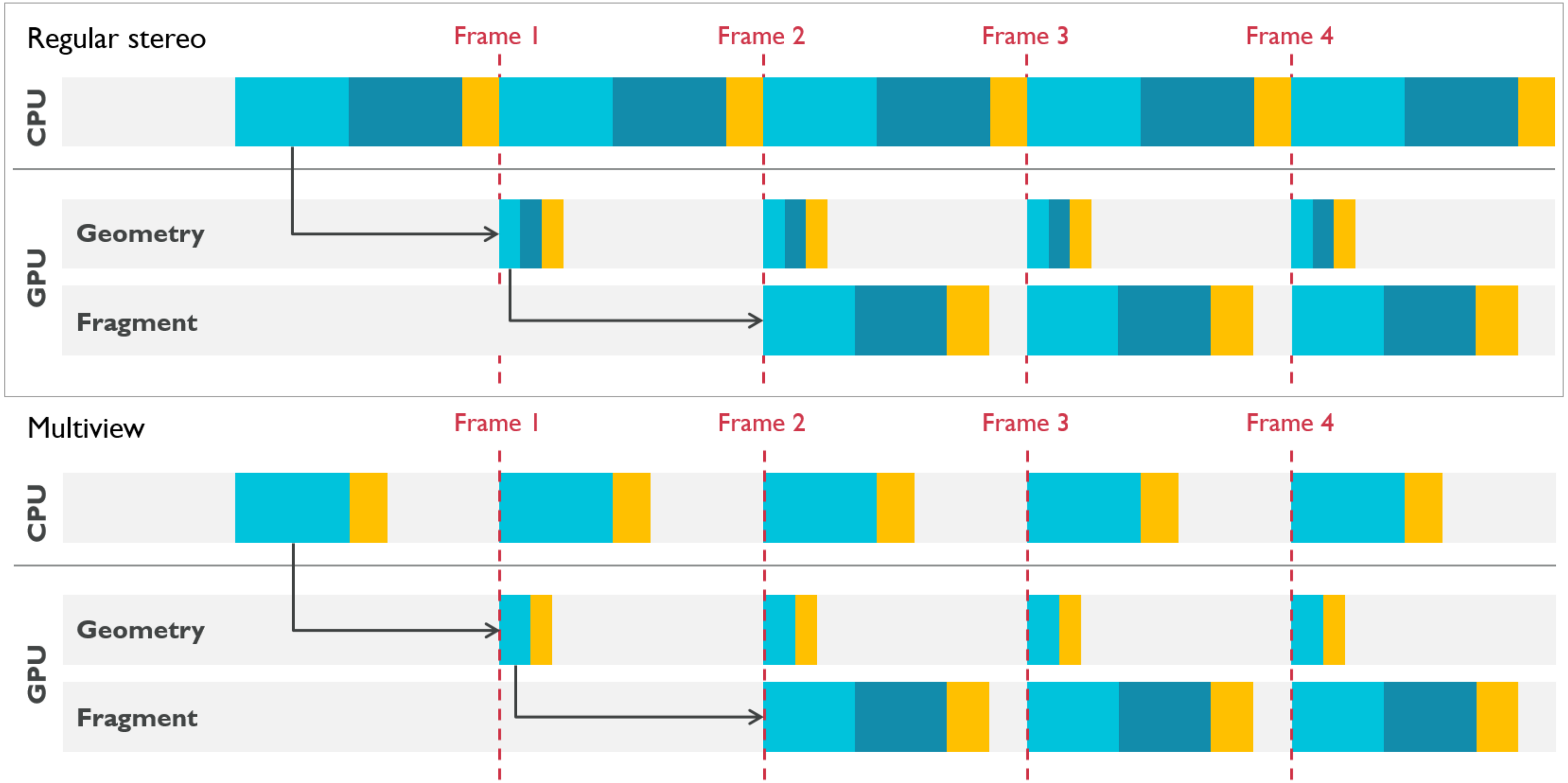


© ARM 2017

Mobile Multiview

Mobile Multiview

- Regular vs. multiview CPU-GPU timeline



© ARM 2017

Mobile Multiview

- Overloaded term
  - Instanced stereo for PC
  - Multiview extension of instanced stereo for PS4
  - Single pass stereo from Nvidia
  - DirectX 11 Multiview extension from AMD
  - Multiview OpenGL ES extension

© ARM 2017

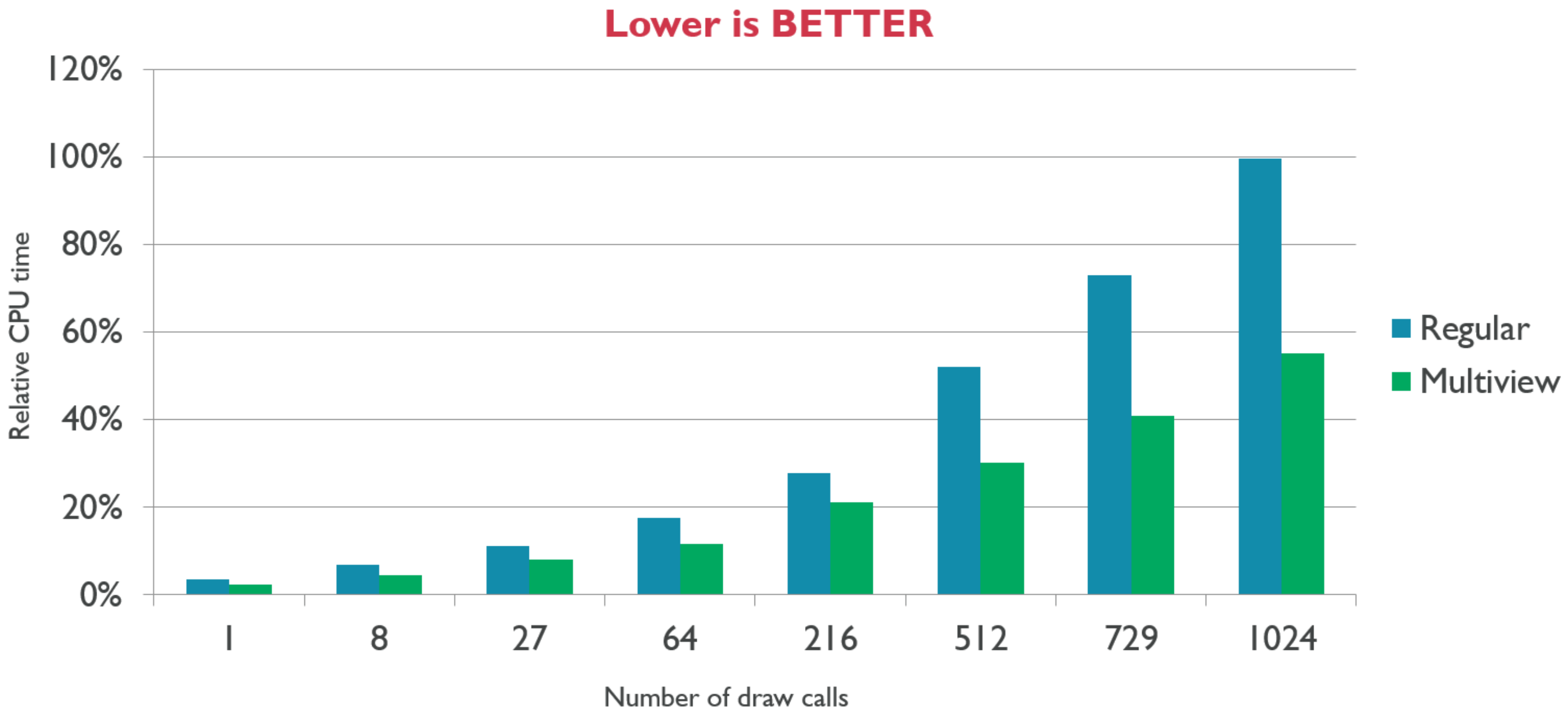
Mobile Multiview

- UE4 implementation
  - PC/PS4 Instanced stereo and PS4 multiview
    - Standard graphics pipeline based
      - Instanced draw call, transform, culling, clipping vertex shader
    - Small extension for PS4 to reduce vertex shader work
  - Mobile multiview
    - Draw call instancing and vertex work done entirely by the driver
    - Leverages view uniform system from instanced stereo

© ARM 2017

Mobile Multiview

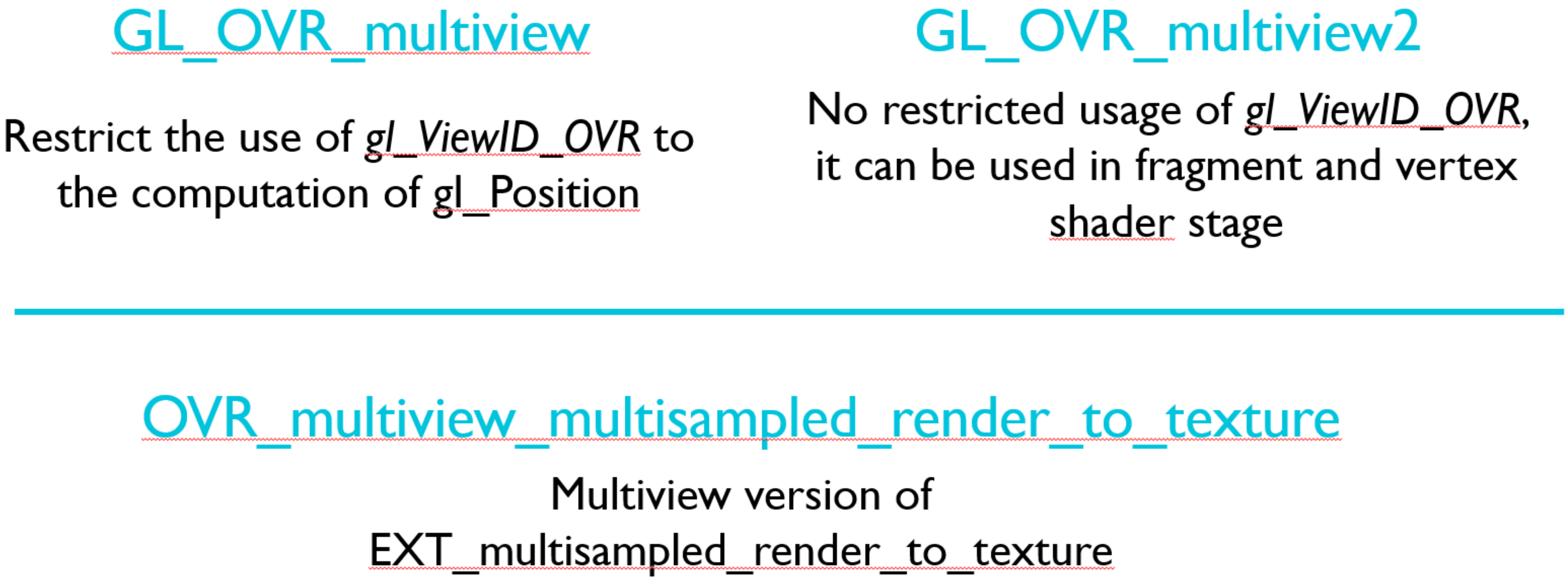
- Multiview: CPU performance



© ARM 2017

Mobile Multiview

- GL_OVR_multiview



GL_OVR_multiview

Restrict the use of *gl_ViewID_OVR* to the computation of gl_Position

GL_OVR_multiview2

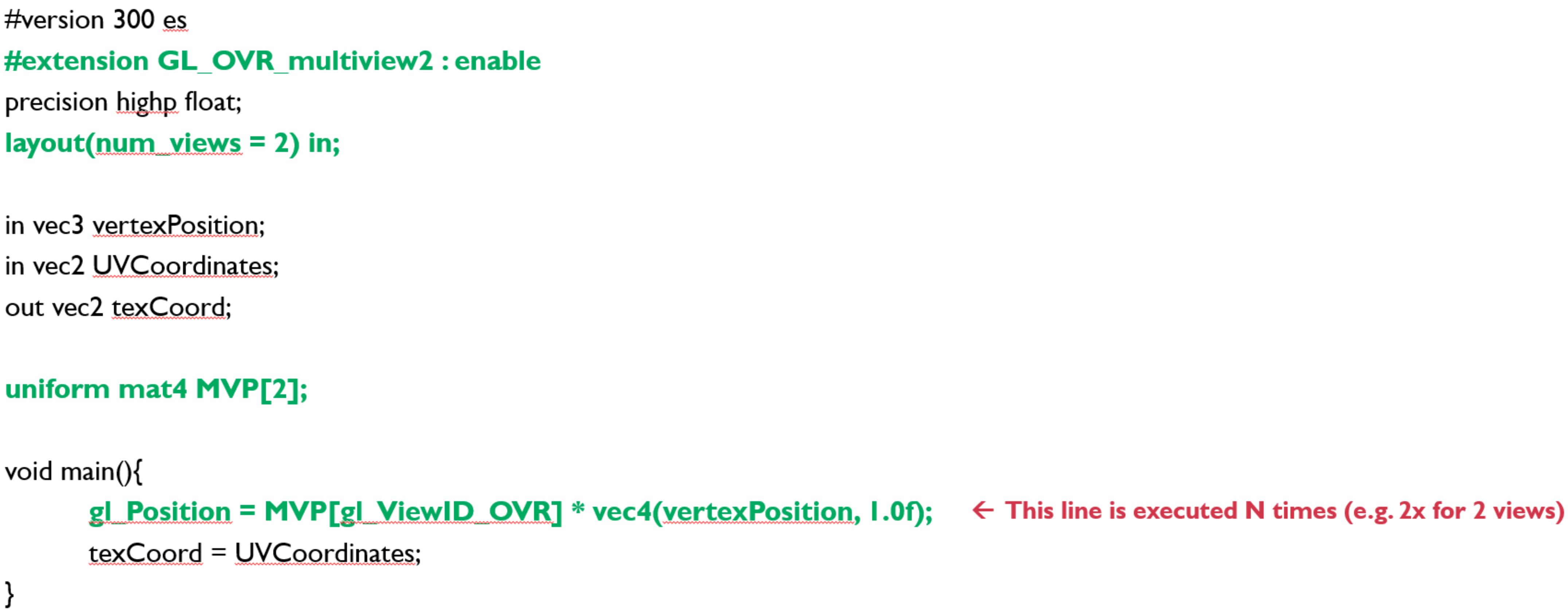
No restricted usage of *gl_ViewID_OVR*, it can be used in fragment and vertex shader stage

OVR_multiview_multisampled_render_to_texture

Multiview version of EXT_multisampled_render_to_texture

Mobile Multiview

- Vertex shader with multiview

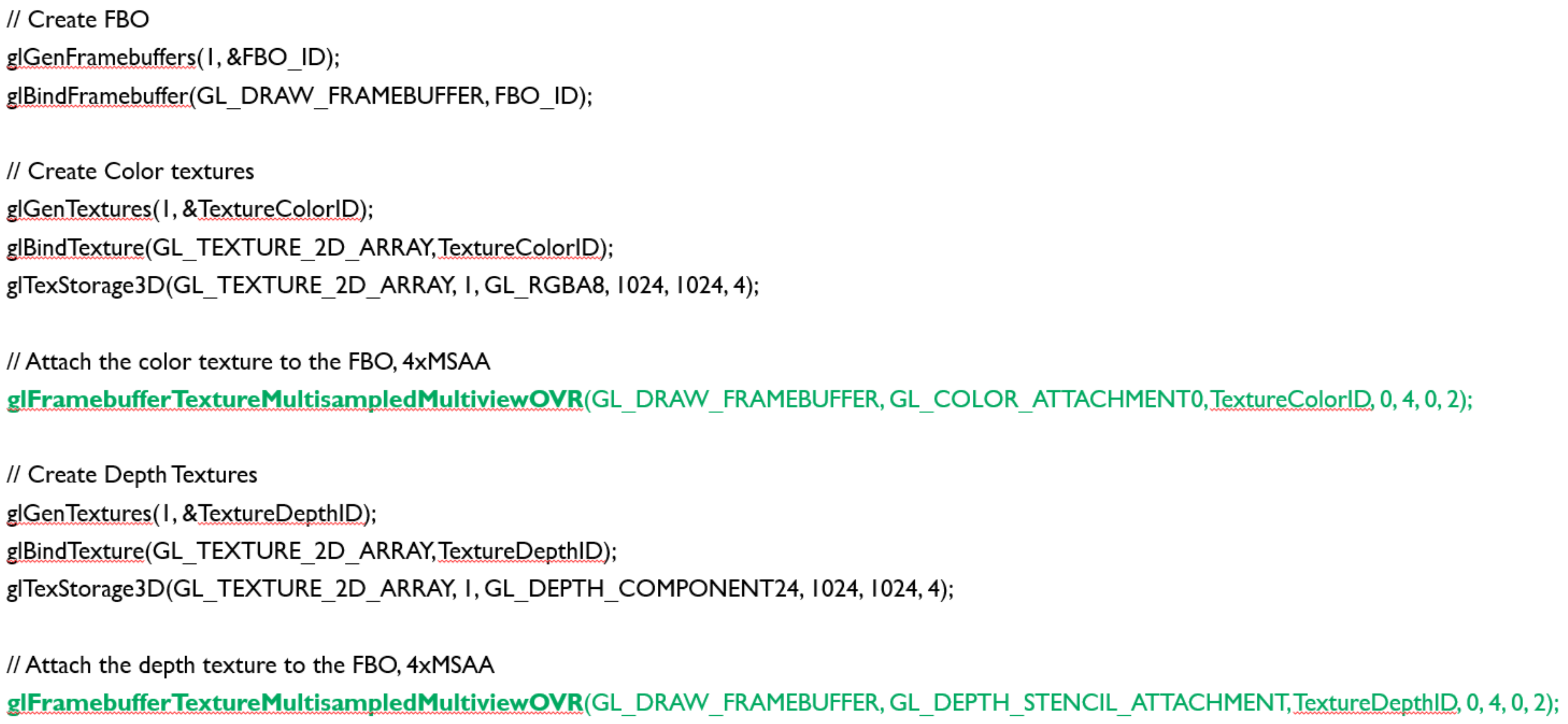


```
#version 300 es
#extension GL_OVR_multiview2 : enable
precision highp float;
layout(num_views = 2) in;

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);    ← This line is executed N times (e.g. 2x for 2 views)
        texCoord = UVCoordinates;
}
```

© ARM 2017

Mobile Multiview

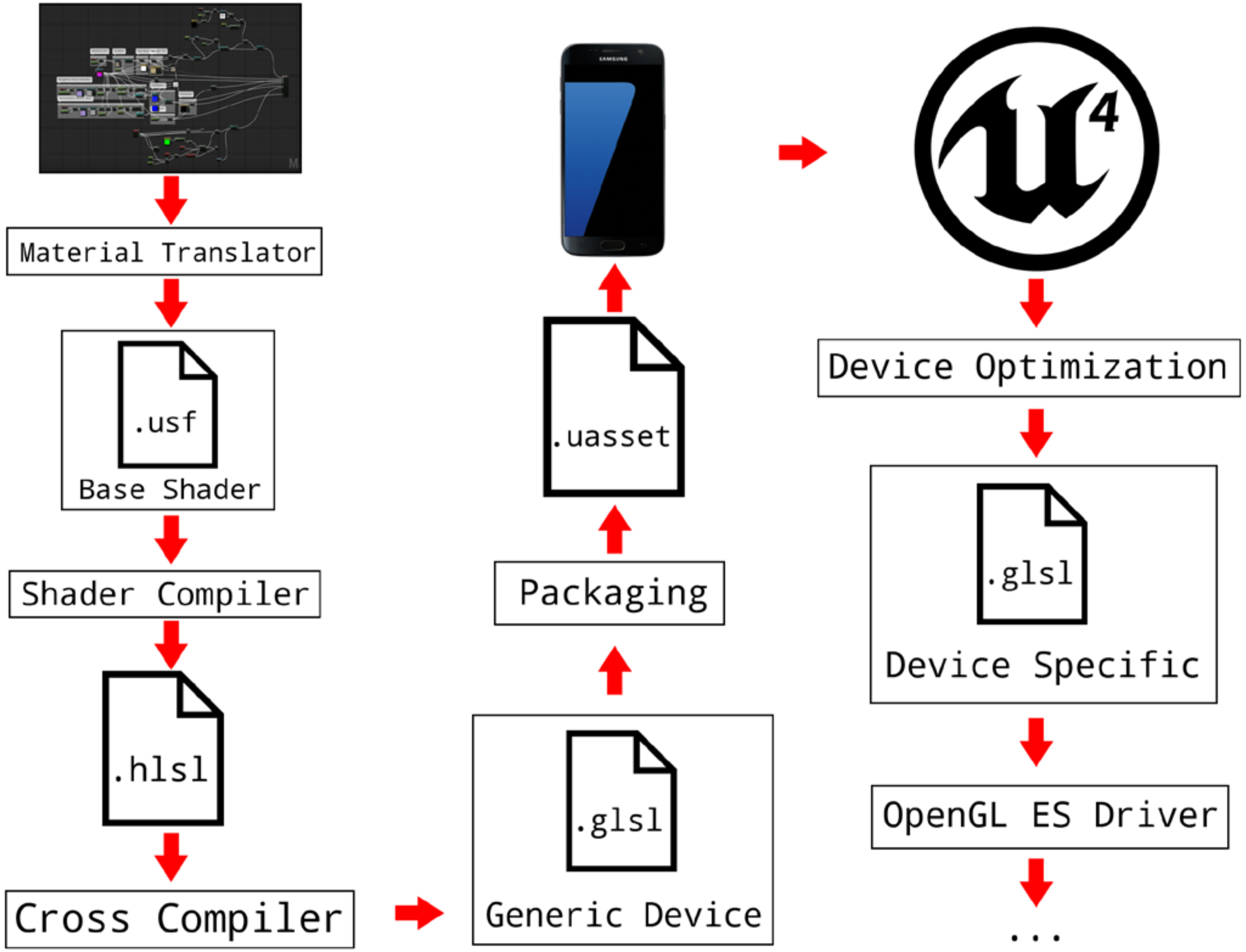
- Using multiview in an application



```
// Create FBO
glGenFramebuffers(1, &FBO_ID);
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, FBO_ID);

// Create Color textures
glGenTextures(1, &TextureColorID);
glBindTexture(GL_TEXTURE_2D_ARRAY,TextureColorID);
glTexStorage3D(GL_TEXTURE_2D_ARRAY, 1, GL_RGBA8, 1024, 1024, 4);

// Attach the color texture to the FBO, 4xMSAA
glFramebufferTextureMultisampledMultiviewOVR(GL_DRAW_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,TextureColorID, 0, 4, 0, 2);

// Create Depth Textures
glGenTextures(1, &TextureDepthID);
glBindTexture(GL_TEXTURE_2D_ARRAY,TextureDepthID);
glTexStorage3D(GL_TEXTURE_2D_ARRAY, 1, GL_DEPTH_COMPONENT24, 1024, 1024, 4);

// Attach the depth texture to the FBO, 4xMSAA
glFramebufferTextureMultisampledMultiviewOVR(GL_DRAW_FRAMEBUFFER, GL_DEPTH_STENCIL_ATTACHMENT,TextureDepthID, 0, 4, 0, 2);
```

© ARM 2017

Mobile Multiview



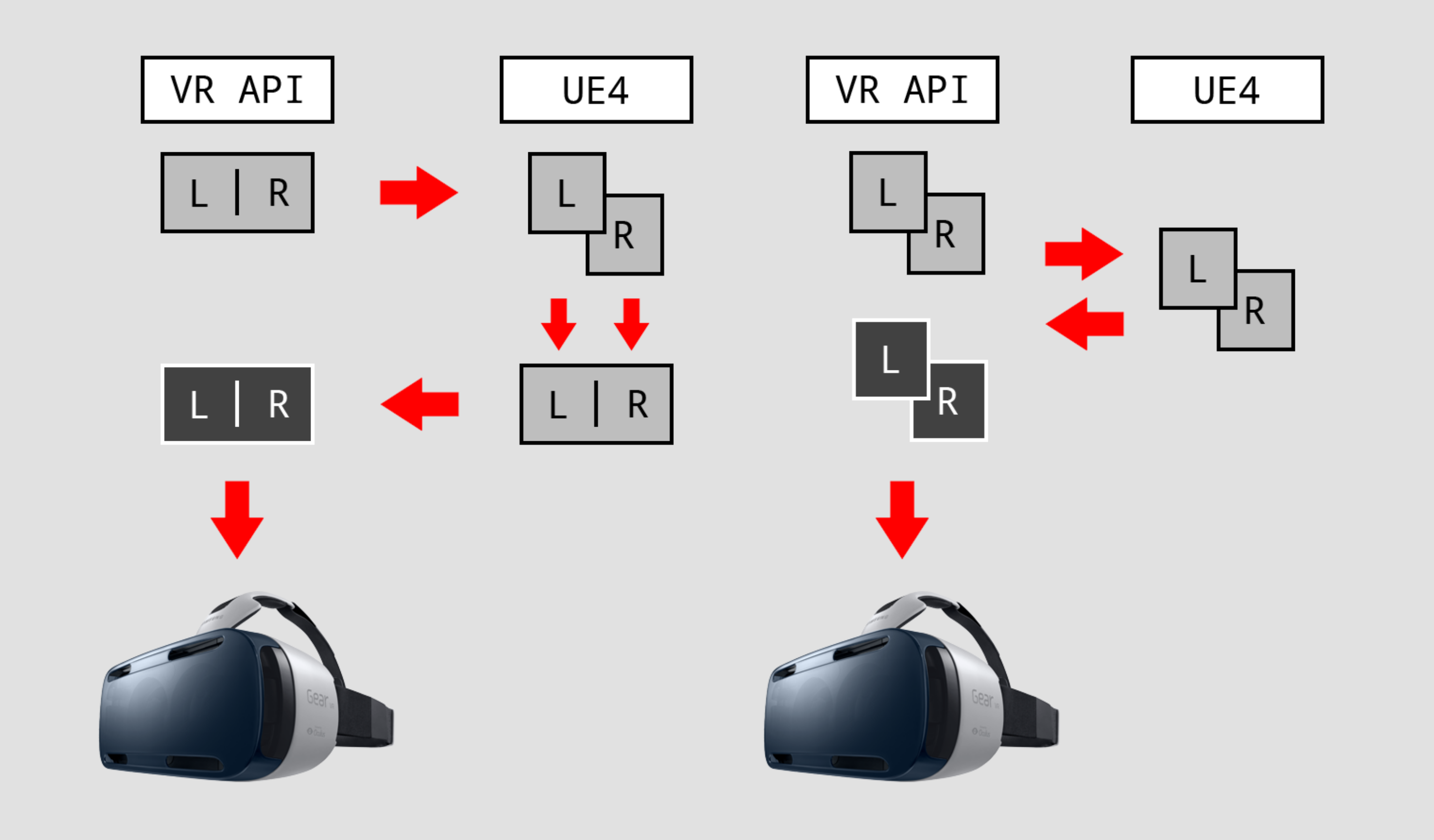
Mobile Multiview

- Driver support landscape
  - Multiple GPU vendors
  - Many driver bugs in initial implementations across all vendors
  - Long delay between driver updates and availability on end user devices
  - We strip out multiview code from the shader during application initialization if the device is known to have issues to ensure driver bugs don't break your application
  - Samsung Galaxy S6, Samsung Galaxy S7 Mali (Android M and N), S7 Adreno (Android N)

© ARM 2017

Mobile Multiview

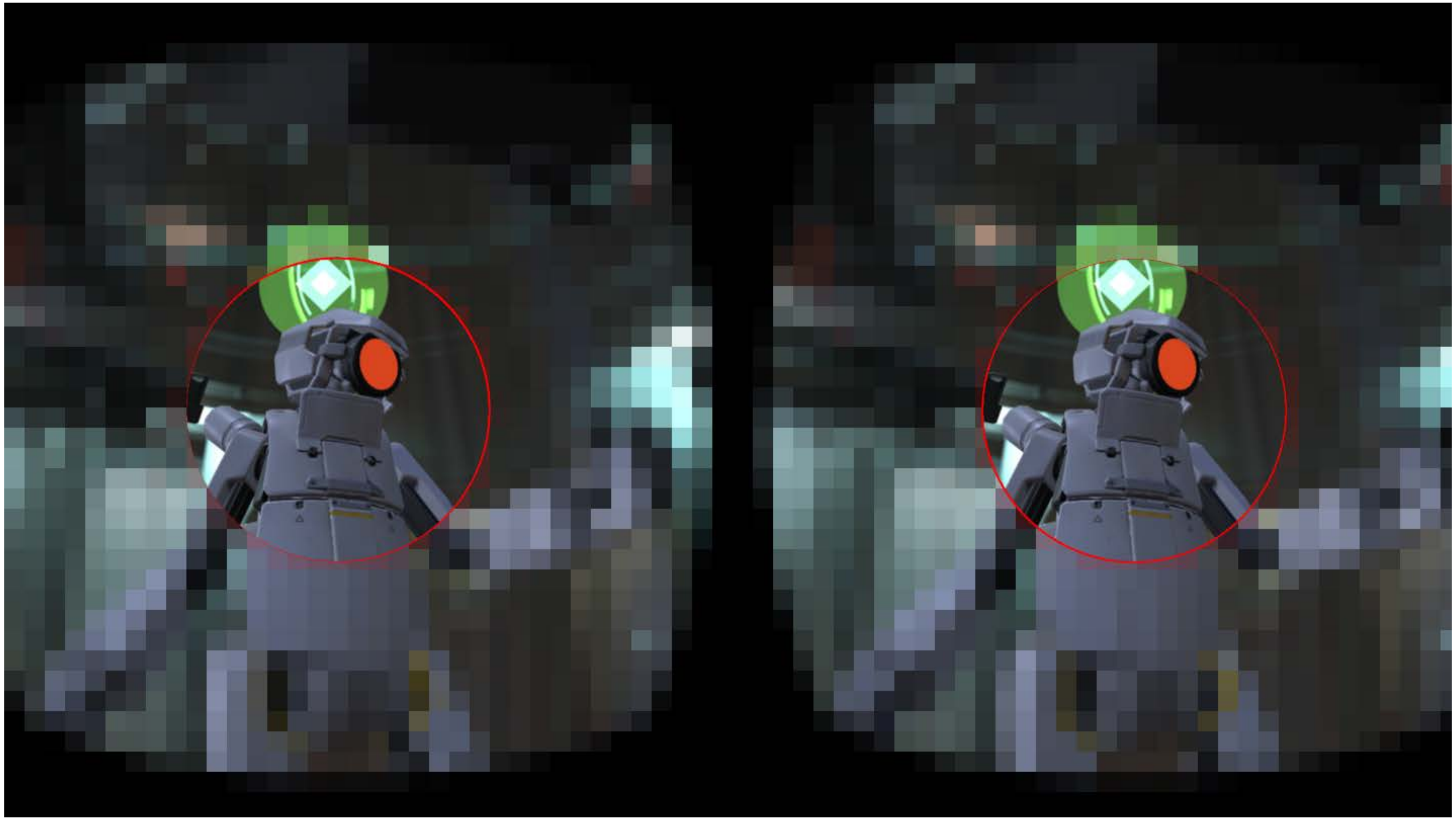
- Current work in development



New technologies in the horizon using Multiview

- Foveated Rendering

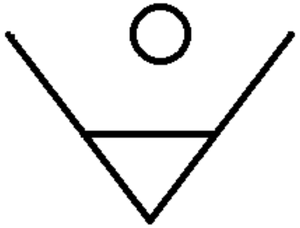
Foveated Rendering



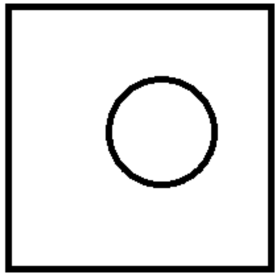
© ARM 2017

Foveated Rendering: 4-view multiview

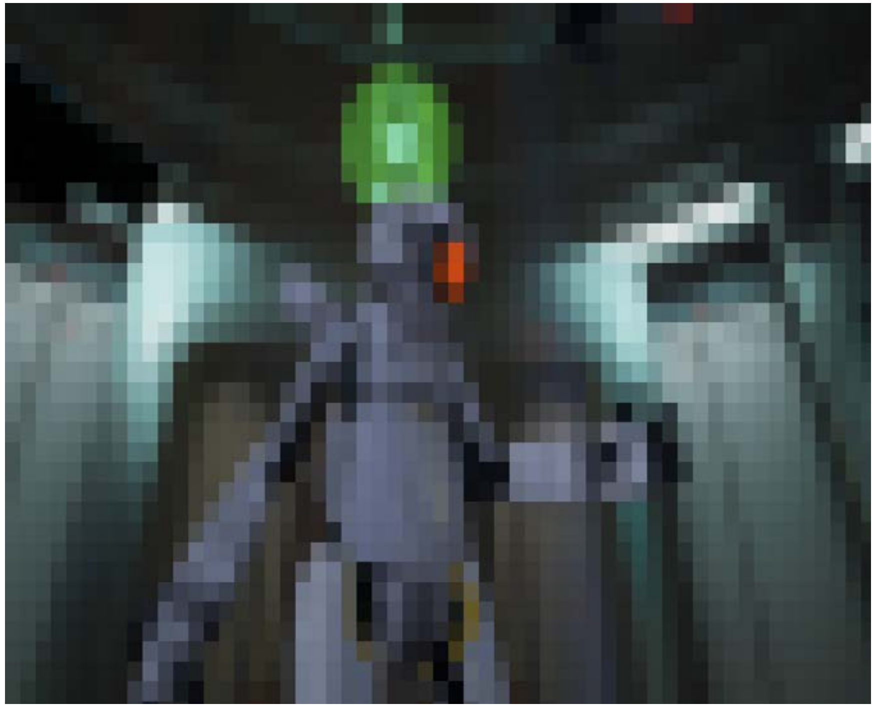
FOV



RT

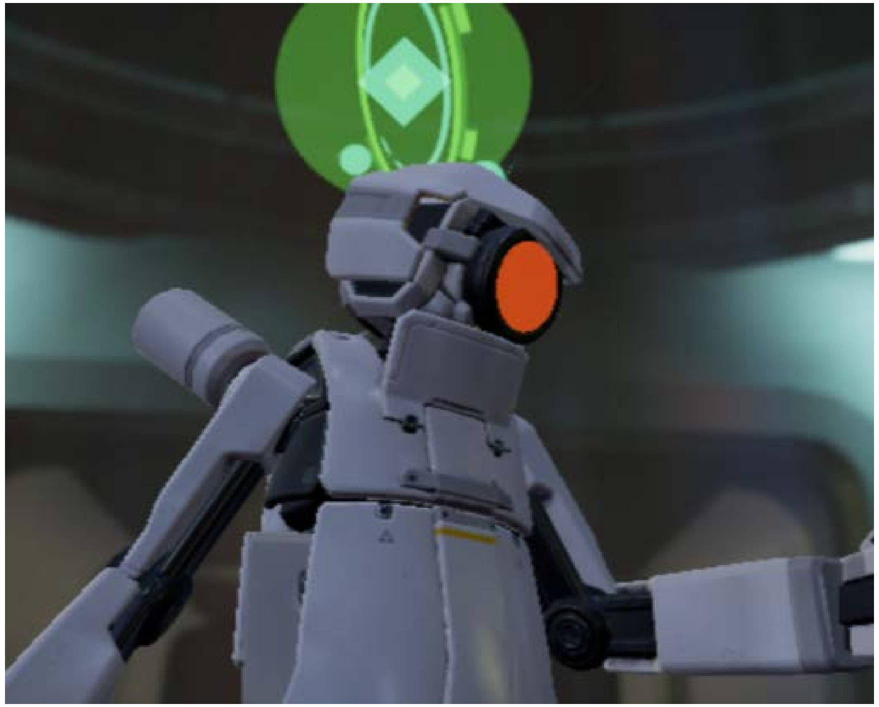


360x360



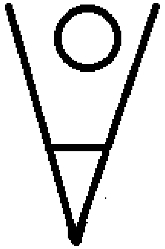
Left eye periphery

360x360

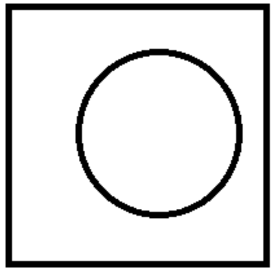


Left eye fovea

FOV



RT



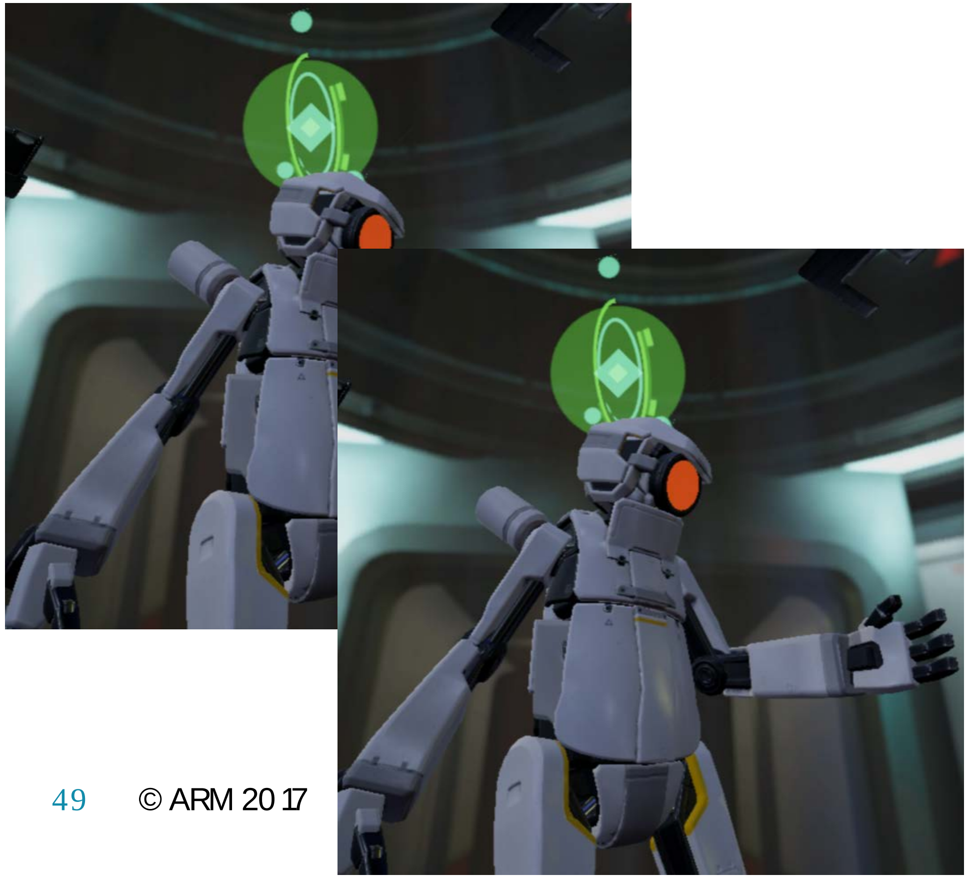
© ARM 2017

Foveated Rendering: 4-view multiview

Pipeline

- Current 2-view Multiview in UE4.14

Scene Rendering into
Texture array of 2x1024x1024 = 2.09 MPx



Blit

Side-by-Size texture 2048x1024

ARM

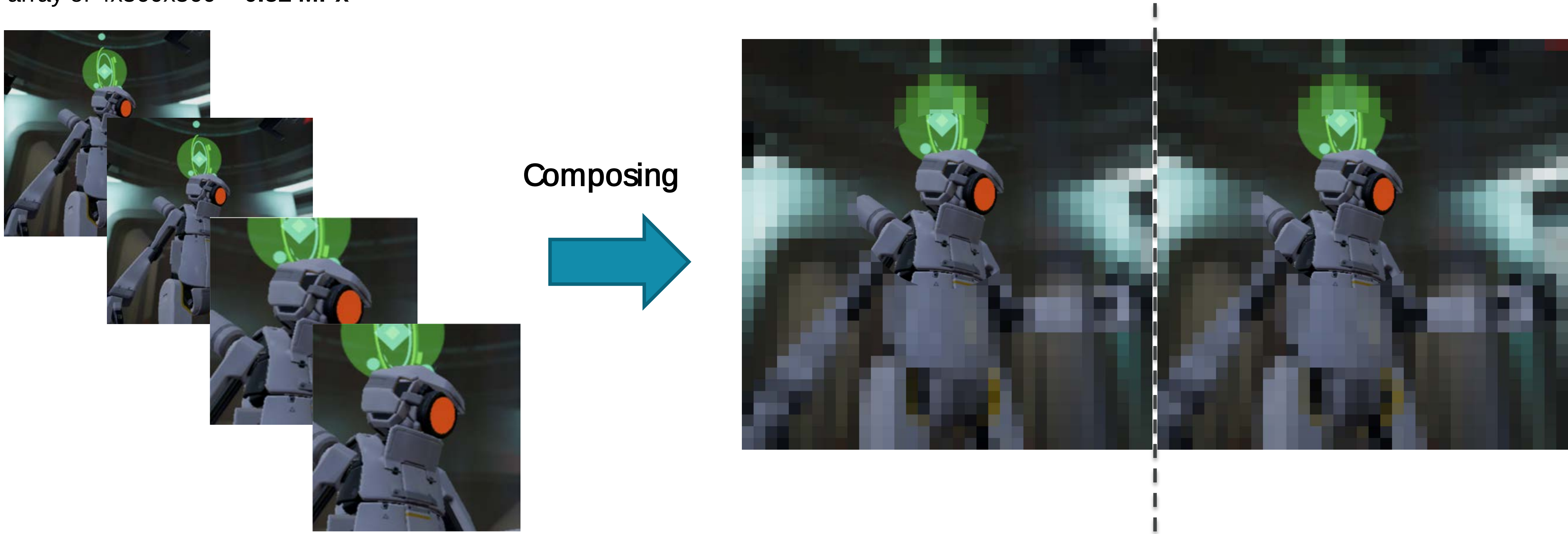
Foveated Rendering: 4-view multiview

Pipeline

- Foveated Rendering with 4-view Multiview (65% reduction)

Scene Rendering into
Texture array of 4x360x360 = **0.52 MPx**

Side-by-Size texture 2048x1024

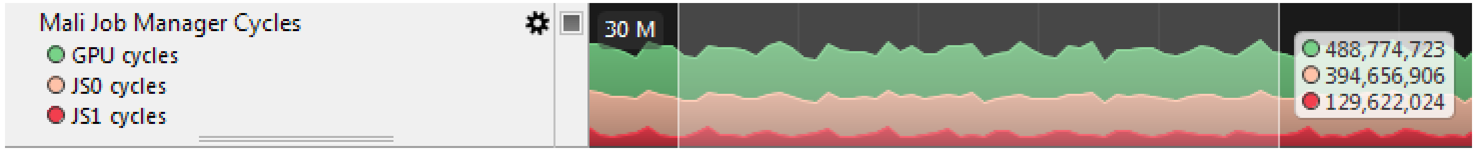


Composing

© ARM 2017

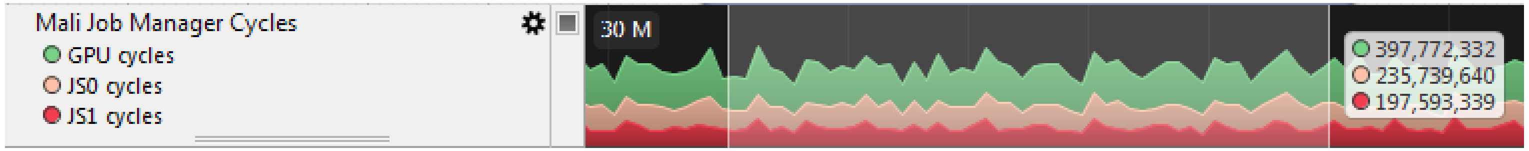
Foveated Rendering: Results

Multiview



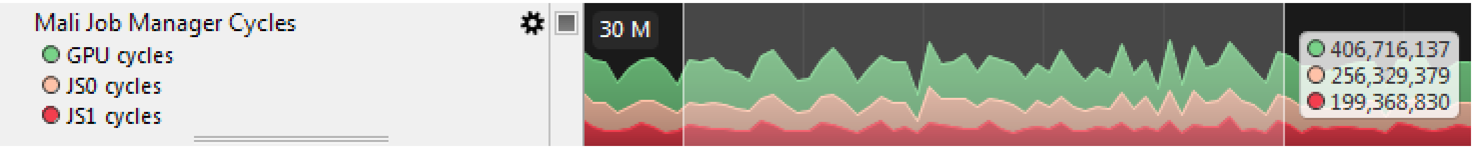
Total: 488 Mcycles
Vertex: 129 Mcycles
Fragment: 394 Mcycles

Foveated (35% original framebuffer size)



Total: 397 Mcycles (-20%)
Vertex: 197 Mcycles (+52%)
Fragment: 235 Mcycles (-40%)

Foveated 8xMSAA (35% original framebuffer size)



Total: 406 Mcycles (-17%)
Vertex: 199 Mcycles (+53%)
Fragment: 256 Mcycles (-35%)

© ARM 2017

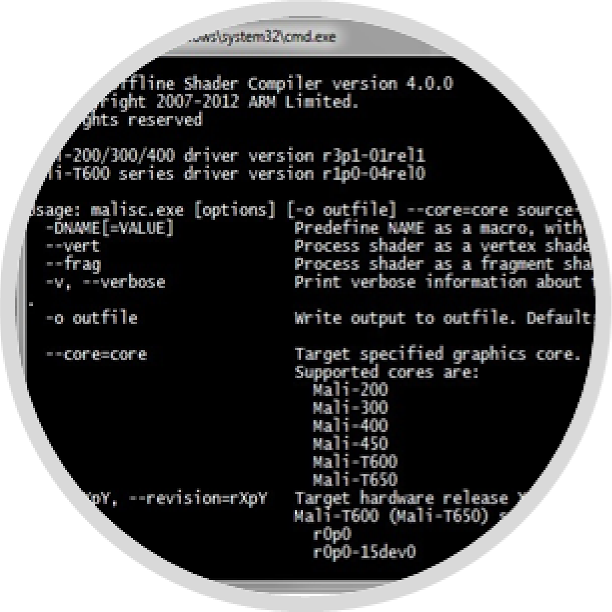
Debugging and Profiling on Mali

- Mali Offline Shader Compiler
- Mali Graphics Debugger (MGD)
- Streamline
  - Result for Foveated Rendering and CircuitVR

© ARM 2017

Mali Offline Shader Compiler

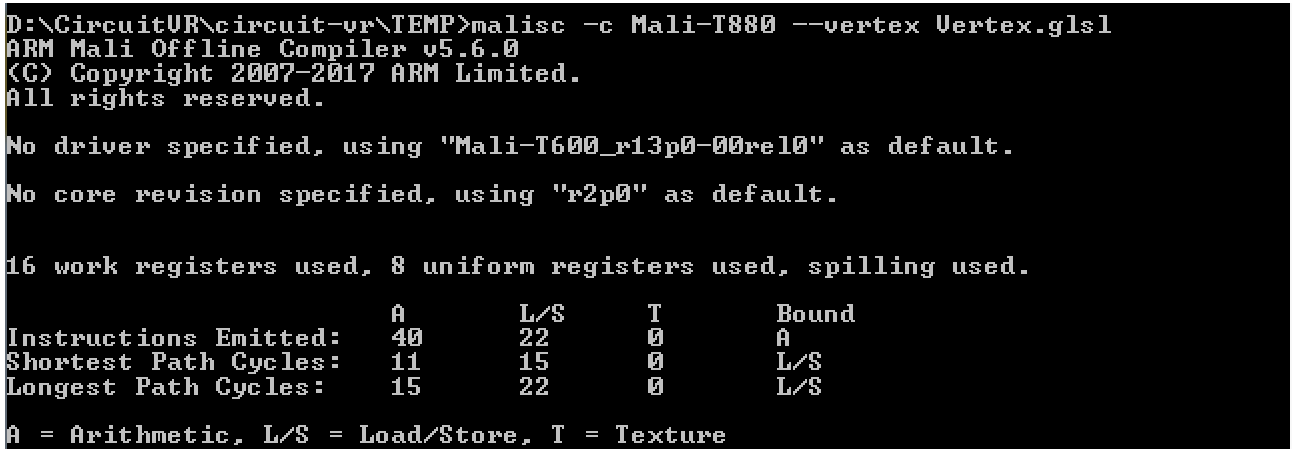
- Mali Offline Compiler
- Analyze shader performance
- Command line tool. Easy to integrate.
- Number of cycles
- Registers utilization



```
ffline Shader Compiler version 4.0.0
right 2007-2012 ARM Limited.
ghts reserved

i-200/300/400 driver version r3p1-01rel1
li-T600 series driver version r1p0-04rel0

sage: malisc.exe [options] [-o outfile] --core=core source
 -DNAME[=VALUE]              Predefine NAME as a macro, with
 --vert                      Process shader as a vertex shade
 --frag                      Process shader as a fragment sha
 -v, --verbose               Print verbose information about

 -o outfile                  Write output to outfile. Default

 --core=core                 Target specified graphics core.
                             Supported cores are:
                               Mali-200
                               Mali-300
                               Mali-400
                               Mali-450
                               Mali-T600
                               Mali-T650
 pY, --revision=rXpY         Target hardware release
                             Mali-T600 (Mali-T650)
                               r0p0
                               r0p0-15dev0
```

Mali Offline Shader Compiler use-case

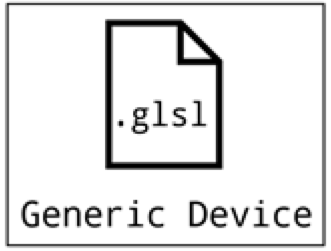
- Shows how many cycles the shortest and longest path takes:
  - Arithmetic pipeline
  - Load/Store pipeline:
  - Texture Pipeline

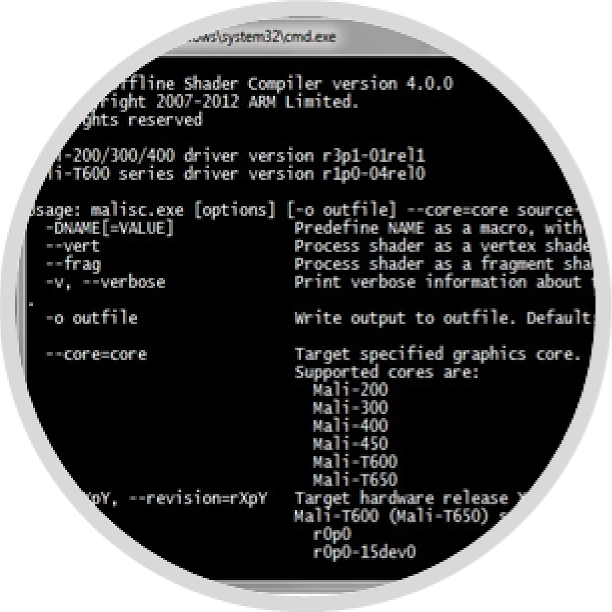


```
D:\CircuitVR\circuit-vr\TEMP>malisc -c Mali-T880 --vertex Vertex.glsl
ARM Mali Offline Compiler v5.6.0
(C) Copyright 2007-2017 ARM Limited.
All rights reserved.

No driver specified, using "Mali-T600_r13p0-00rel0" as default.

No core revision specified, using "r2p0" as default.


16 work registers used, 8 uniform registers used, spilling used.

                        A       L/S     T       Bound
Instructions Emitted:   40      22      0       A
Shortest Path Cycles:   11      15      0       L/S
Longest Path Cycles:    15      22      0       L/S

A = Arithmetic, L/S = Load/Store, T = Texture
```

Mali Offline Shader Compiler: Getting the UE4 shaders

- Enable shader dumps and shader development in ConsoleVariables.ini
- Invalidate the shader cache with r.InvalidataCachedShaders 1
- Restart the editor
  - This will dump the shaders in <ProjectFolder>/Saved/ShaderDebugInfo/PCD3D_SM[4|5]
- To generate the shaders for mobile:
  - Package the game for a mobile platform or
  - Activate the mobile preview
- Shaders will be in
  - <ProjectFolder>/Saved/ShaderDebugInfo/GLSL_ES[2|3]





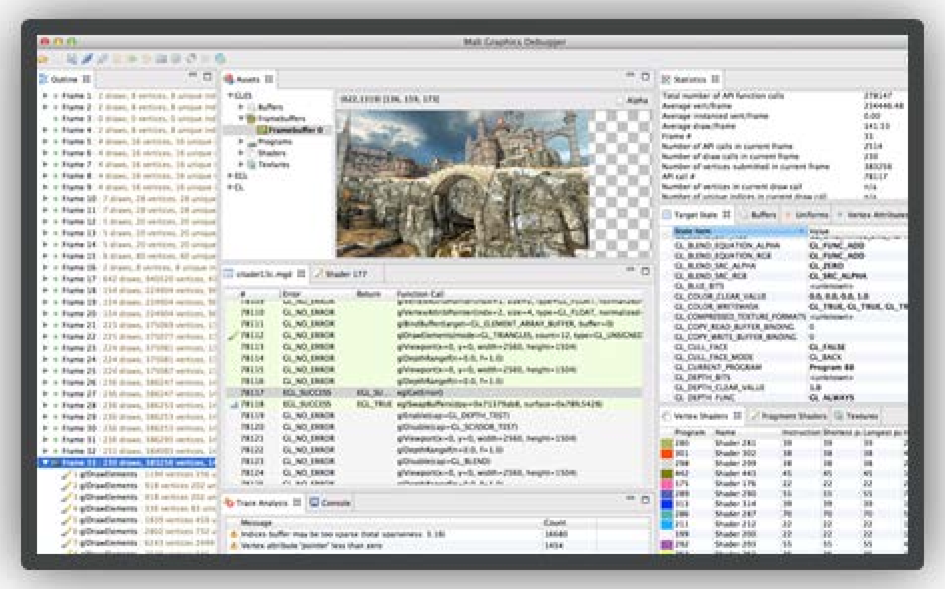
```
ffline Shader Compiler version 4.0.0
right 2007-2012 ARM Limited.
ghts reserved

-200/300/400 driver version r3p1-01rel1
li-T600 series driver version r1p0-04rel0

sage: malisc.exe [options] [-o outfile] --core=core source
  -DNAME[=VALUE]             Predefine NAME as a macro, with
  --vert                     Process shader as a vertex shade
  --frag                     Process shader as a fragment sha
  -v, --verbose              Print verbose information about

  -o outfile                 Write output to outfile. Default

  --core=core                Target specified graphics core.
                             Supported cores are:
                               Mali-200
                               Mali-300
                               Mali-400
                               Mali-450
                               Mali-T600
                               Mali-T650
pY, --revision=rXpY          Target hardware release
                             Mali-T600 (Mali-T650)
                               r0p0
                               r0p0-15dev0
```

Mali Graphics Debugger (MGD)

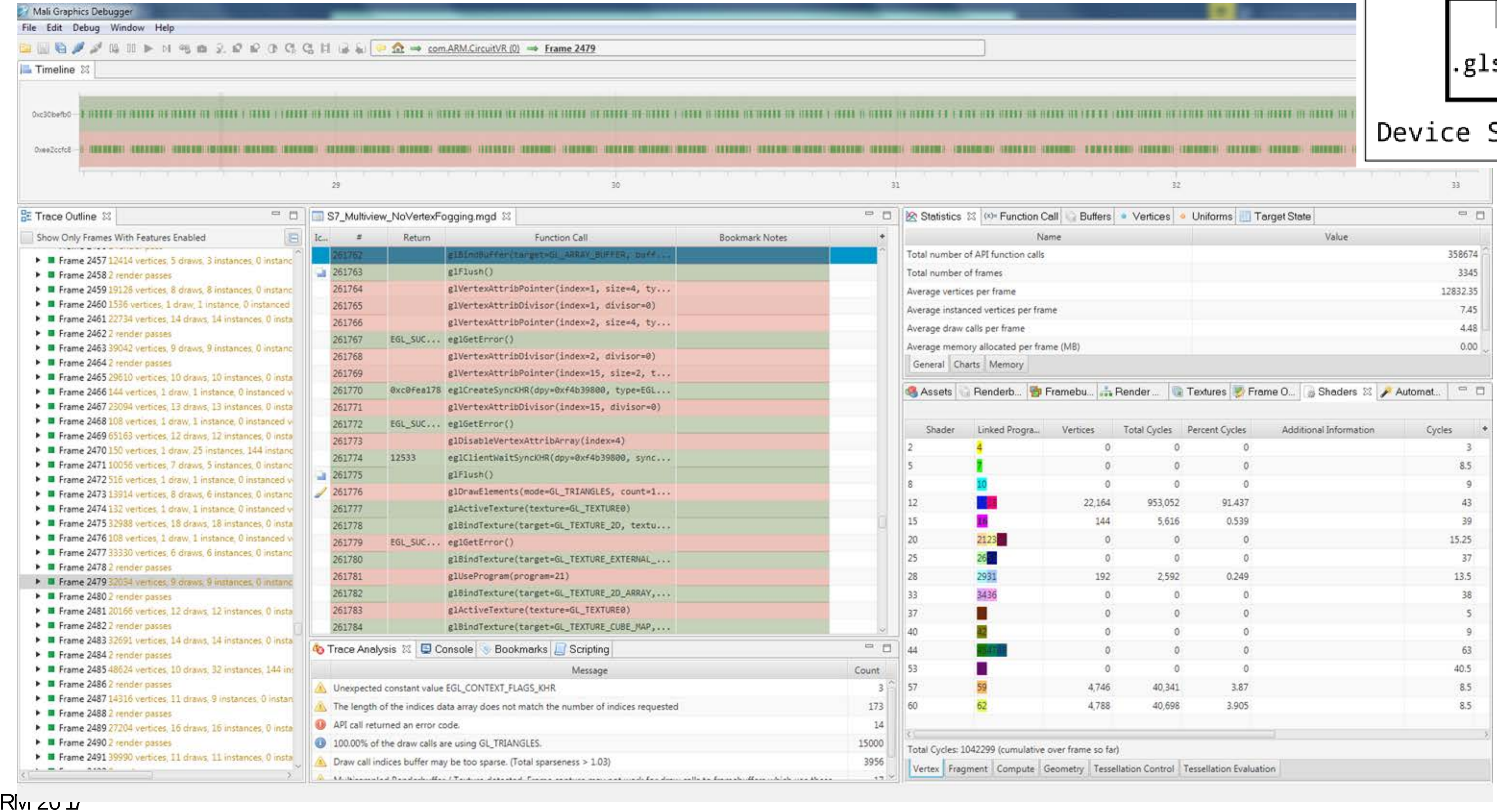
- Runtime API Trace and resources analysis
- OpenGL ES, OpenCL
- Debug and improve performance at frame level

- Available in UE4.15!! ☺

- But currently not fully working with VR ☹
- Still useful to debug a No-VR version



Mali Graphics Debugger (MGD)


Mali Graphics Debugger
File Edit Debug Window Help
com.ARM.CircuitVR (0) → Frame 2479
Timeline
0xc30befb0
0xee2ccfc8
29
30
31
32
33

Timeline

Drawcalls

Trace Outline

☐ Show Only Frames With Features Enabled

- ▸ Frame 2457 12414 vertices, 5 draws, 3 instances, 0 instanc
- ▸ Frame 2458 2 render passes
- ▸ Frame 2459 19128 vertices, 8 draws, 8 instances, 0 instanc
- ▸ Frame 2460 1536 vertices, 1 draw, 1 instance, 0 instanced
- ▸ Frame 2461 22734 vertices, 14 draws, 14 instances, 0 insta
- ▸ Frame 2462 2 render passes
- ▸ Frame 2463 39042 vertices, 9 draws, 9 instances, 0 instanc
- ▸ Frame 2464 2 render passes
- ▸ Frame 2465 29610 vertices, 10 draws, 10 instances, 0 insta
- ▸ Frame 2466 144 vertices, 1 draw, 1 instance, 0 instanced v
- ▸ Frame 2467 23094 vertices, 13 draws, 13 instances, 0 insta
- ▸ Frame 2468 108 vertices, 1 draw, 1 instance, 0 instanced v
- ▸ Frame 2469 65163 vertices, 12 draws, 12 instances, 0 insta
- ▸ Frame 2470 150 vertices, 1 draw, 25 instances, 144 instanc
- ▸ Frame 2471 10056 vertices, 7 draws, 5 instances, 0 instanc
- ▸ Frame 2472 516 vertices, 1 draw, 1 instance, 0 instanced v
- ▸ Frame 2473 13914 vertices, 8 draws, 6 instances, 0 instanc
- ▸ Frame 2474 132 vertices, 1 draw, 1 instance, 0 instanced v
- ▸ Frame 2475 32988 vertices, 18 draws, 18 instances, 0 insta
- ▸ Frame 2476 108 vertices, 1 draw, 1 instance, 0 instanced v
- ▸ Frame 2477 33330 vertices, 6 draws, 6 instances, 0 instanc
- ▸ Frame 2478 2 render passes
- ▸ Frame 2479 32034 vertices, 9 draws, 9 instances, 0 instanc
- ▸ Frame 2480 2 render passes
- ▸ Frame 2481 20166 vertices, 12 draws, 12 instances, 0 insta
- ▸ Frame 2482 2 render passes
- ▸ Frame 2483 32691 vertices, 14 draws, 14 instances, 0 insta
- ▸ Frame 2484 2 render passes
- ▸ Frame 2485 48624 vertices, 10 draws, 32 instances, 144 ins
- ▸ Frame 2486 2 render passes
- ▸ Frame 2487 14316 vertices, 11 draws, 9 instances, 0 instan
- ▸ Frame 2488 2 render passes
- ▸ Frame 2489 27204 vertices, 16 draws, 16 instances, 0 insta
- ▸ Frame 2490 2 render passes
- ▸ Frame 2491 39990 vertices, 11 draws, 11 instances, 0 insta

S7_Multiview_NoVertexFogging.mgd

Ic...	#	Return	Function Call	Bookmark Notes
	261762		glBindBuffer(target=GL_ARRAY_BUFFER, buff...	
	261763		glFlush()	
	261764		glVertexAttribPointer(index=1, size=4, ty...	
	261765		glVertexAttribDivisor(index=1, divisor=0)	
	261766		glVertexAttribPointer(index=2, size=4, ty...	
	261767	EGL_SUC...	eglGetError()	
	261768		glVertexAttribDivisor(index=2, divisor=0)	
	261769		glVertexAttribPointer(index=15, size=2, t...	
	261770	0xc0fea178	eglCreateSyncKHR(dpy=0xf4b39800, type=EGL...	
	261771		glVertexAttribDivisor(index=15, divisor=0)	
	261772	EGL_SUC...	eglGetError()	
	261773		glDisableVertexAttribArray(index=4)	
	261774	12533	eglClientWaitSyncKHR(dpy=0xf4b39800, sync...	
	261775		glFlush()	
	261776		glDrawElements(mode=GL_TRIANGLES, count=1...	
	261777		glActiveTexture(texture=GL_TEXTURE0)	
	261778		glBindTexture(target=GL_TEXTURE_2D, textu...	
	261779	EGL_SUC...	eglGetError()	
	261780		glBindTexture(target=GL_TEXTURE_EXTERNAL_...	
	261781		glUseProgram(program=21)	
	261782		glBindTexture(target=GL_TEXTURE_2D_ARRAY,...	
	261783		glActiveTexture(texture=GL_TEXTURE0)	
	261784		glBindTexture(target=GL_TEXTURE_CUBE_MAP,...	

Trace Analysis | Console | Bookmarks | Scripting

Message	Count
⚠ Unexpected constant value EGL_CONTEXT_FLAGS_KHR	3
⚠ The length of the indices data array does not match the number of indices requested	173
❗ API call returned an error code.	14
ℹ 100.00% of the draw calls are using GL_TRIANGLES.	15000
⚠ Draw call indices buffer may be too sparse. (Total sparseness > 1.03)	3956

Statistics | Function Call | Buffers | Vertices | Uniforms | Target State

Name	Value
Total number of API function calls	358674
Total number of frames	3345
Average vertices per frame	12832.35
Average instanced vertices per frame	7.45
Average draw calls per frame	4.48
Average memory allocated per frame (MB)	0.00

General | Charts | Memory

Assets | Renderb... | Framebu... | Render ... | Textures | Frame O... | Shaders | Automat...

Shader	Linked Progra...	Vertices	Total Cycles	Percent Cycles	Additional Information	Cycles
2	4	0	0	0		3
5	7	0	0	0		8.5
8	10	0	0	0		9
12	[illegible]	22,164	953,052	91.437		43
15	16	144	5,616	0.539		39
20	2123	0	0	0		15.25
25	26	0	0	0		37
28	2931	192	2,592	0.249		13.5
33	3436	0	0	0		38
37		0	0	0		5
40	42	0	0	0		9
44	[illegible]	0	0	0		63
53		0	0	0		40.5
57	59	4,746	40,341	3.87		8.5
60	62	4,788	40,698	3.905		8.5

Total Cycles: 1042299 (cumulative over frame so far)

Vertex | Fragment | Compute | Geometry | Tessellation Control | Tessellation Evaluation

.glsl

Device Specific

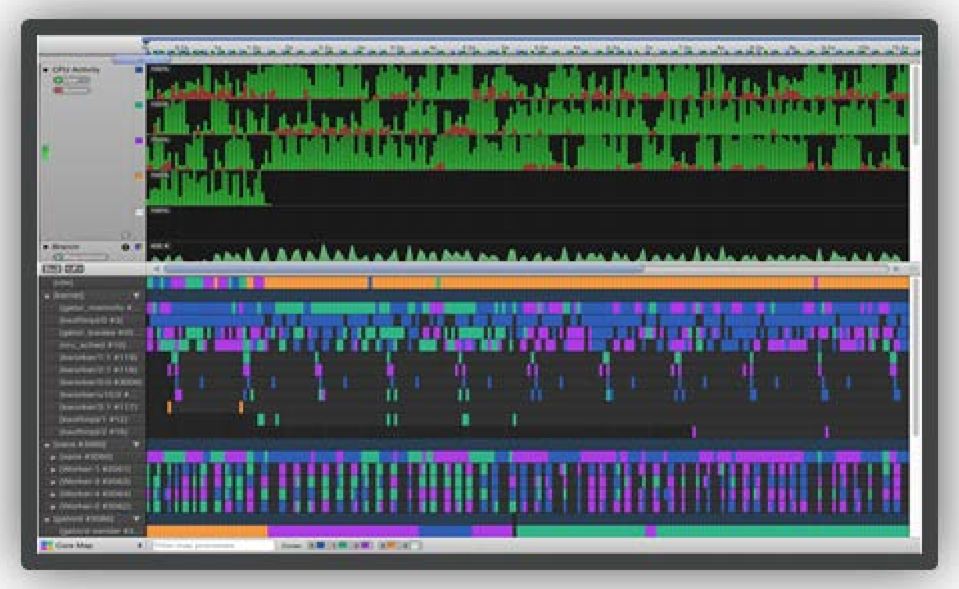
-Statistics
-Buffers
-Vertices
-Uniforms
-TargetState

-Shaders
-FBs
-Textures
-RBs
-Assets
-Etc...

Streamline

- Profile CPUs and Mali GPUs
- Timeline
- HW Counters
- OpenCL visualizer

- New version in April which shows Mali counters without rooting the device.
  - Rooting needed for precise CPU load analysis

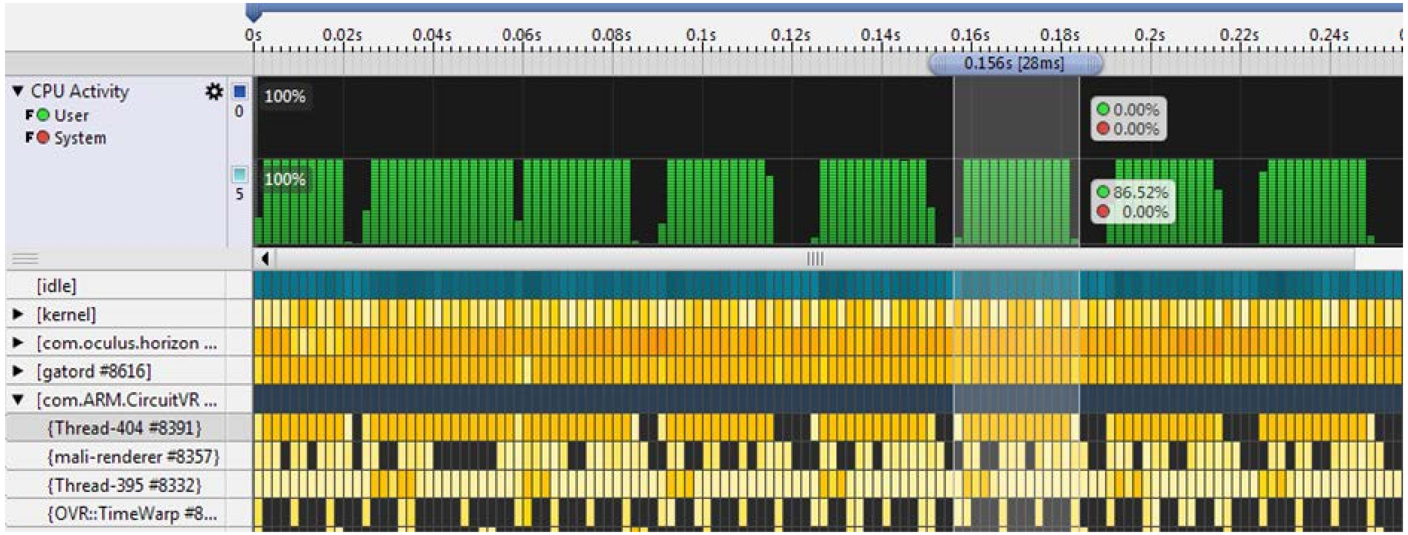


Streamline use-case: Multiview On/Off CPU

- No-Multiview

UE4 Render thread:

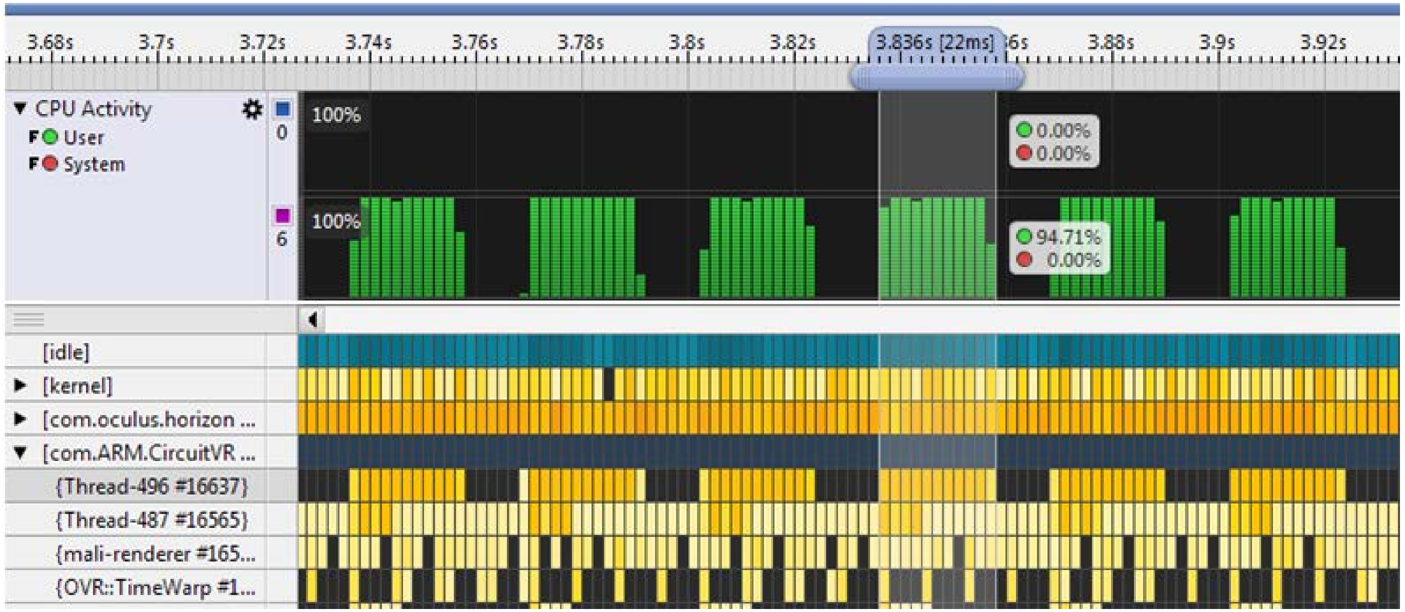
28ms



- Multiview

UE4 Render thread:

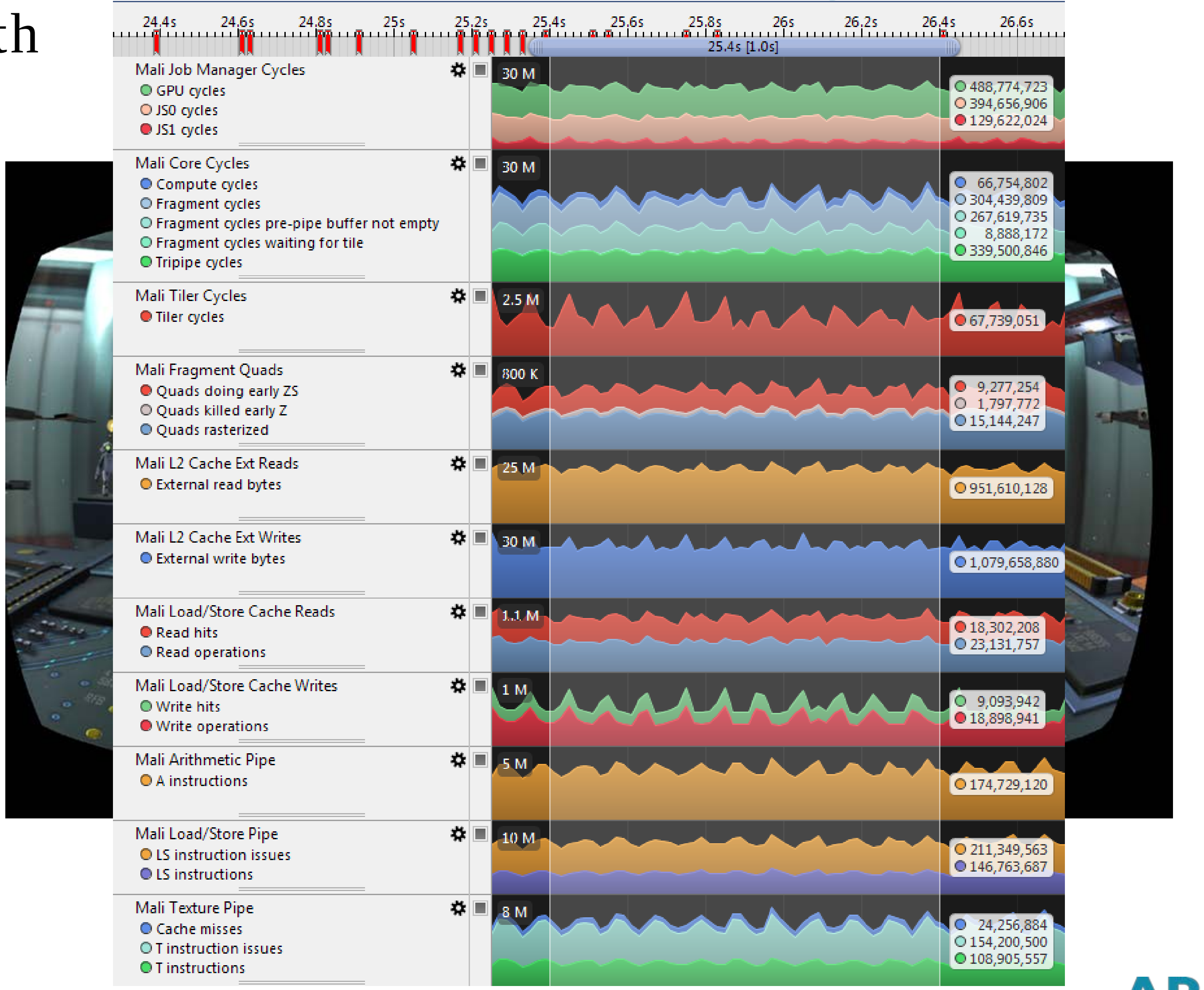
22ms (~23% reduction in CPU load)



© ARM 2017

Streamline use-case: Foveated rendering

- Useful metrics measurable with streamline:
  - GPU active cycles
    - Separated for vertex/fragme
  - Overdraw/Early-Z
  - Bandwidth
  - Cache hit/miss for textures and load/store
  - GPU Utilization
    - Separated for Arith, L/S and Texture
  - Average CPI, Cycles per vertex/fragment
  - Much more!



© ARM 2017

Streamline: GPU references

Description and optimization tips for tiled based gpus:
https://community.arm.com/graphics/b/blog/posts/the-mali-gpu-an-abstract-machine-part-1---frame-pipelining

Description of GPU counter available:
Midgard
https://community.arm.com/graphics/b/documents/posts/mali-midgard-family-performance-counters
Bifrost
https://community.arm.com/graphics/b/documents/posts/mali-bifrost-family-performance-counters

ARM

Q&A

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

Copyright © 2017 ARM Limited