arm

SKULLFISH STUDIOS

Know your limits

Understanding GPU Budgets in Mobile Game Development

Roberto Lopez Mendez (Arm)
Rafael Ferrari (Skullfish Studios)

Agenda

- Mobile VR Optimization & Best practices (Roberto)
  - GPU processing budget approach
  - MGD integration in Unity
  - Single-pass stereo rendering vs multi-pass
  - Mobile VR best practices (MSAA, ASTC, rendering techniques based on local cubemaps)
  - New features coming to VR
- Lila’s Tale experience from Skullfish Studios (Rafael)
  - Game planning, GPU budget
  - Game optimization, MGD
  - Single-pass vs multi-pass stereo rendering: draw calls, power measurement
  - 4xMSAA & ASTC
- Wrap up (Roberto)

GPU processing budget approach

arm

Calculating the GPU processing budget

GPU budget: cycles/frame/pixel; cycles/frame/vertex

Samsung Galaxy S7: octa-core Arm CPU and Arm Mali-T880 MP12 GPU

- 12 GPU cores @ 650 MHz – 12 x 650 M single cycle operations per second
- Target: 60 FPS | Full HD 1920 × 1080 = 2 073 600 pixels.

fragCycleBudget = (130 M cycles/frame) / (2.07 M pixels) ~ 63 cycles/frame/pixel.

fragCycleBudget = (No of GPU fragment cores)(gpuFrequency in Hz) / (FPS*numPixels)*

vertCycleBudget = (No of GPU vertex cores)(gpuFrequency in Hz) / (FPS*numVertices)*

Mali Graphics Debugger integration in Unity

arm

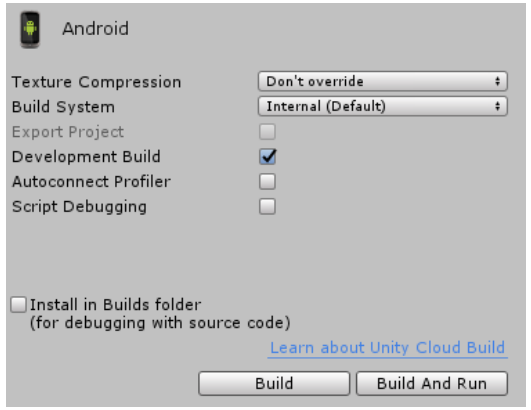
Mali Graphics Debugger (MGD)

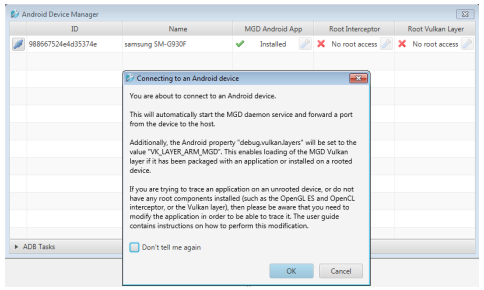
- **Draw-call by Draw-call stepping**
  To identify draw call related issues, redundant draw calls and other opportunities to optimize
- **Texture View**
  Visualize an application's texture usage, to identify opportunities to compress textures or change formats
- **Shader Statistics**
  Understand which 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

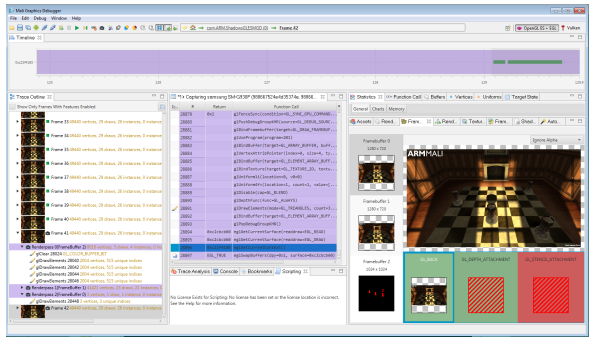
Download MGD for free at developer.arm.com

Mali Graphics Debugger

OpenGL ES

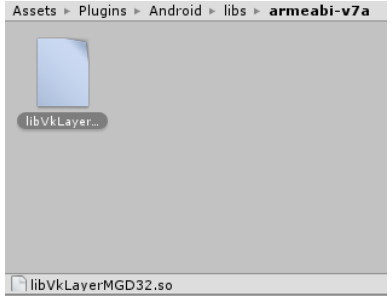




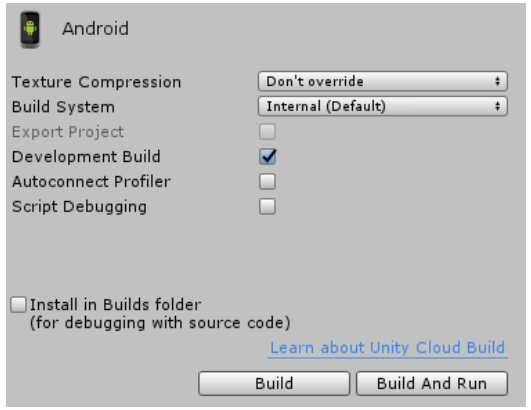


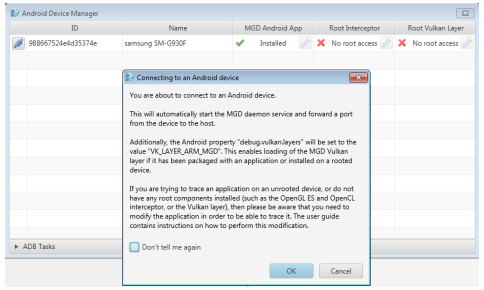
Vulkan

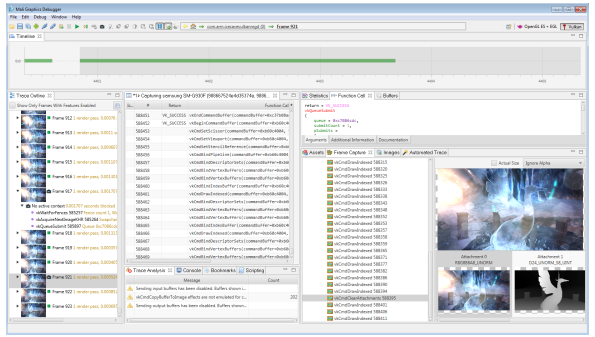
Assets/Plugins/Android/libs/armeabi-v7a/



libVkLayerMGD32.so







Get your vertex & fragment cycle count from MGD

Assets | Renderbuff... | Framebuffe... | Textures | Frame Ove... | Shaders | Automated ...

Shader	Linked Prog...	Vertices	Total Cycles	Percent Cycles	Cycles	A	L/S	T	Uniform Reg...	Work Reg...
355	357	95,856	1,246,128	35.92	13	20	13	0	14	6
352	[illegible]	35,904	753,984	21.734	21	31	21	0	16	8
364	366	16,893	473,004	13.635	28	36	28	0	16	8
367	369	12,918	335,868	9.682	26	41	26	0	16	8
307	309	72,054	259,395	7.477	3.6	11	3	0	10	2
349	351	12,240	220,320	6.351	18	23	18	0	17	7
370	372	9,000	54,000	1.557	6	12	6	0	14	3
304	306	4,836	29,016	0.836	6	13	6	0	16	3

Total Cycles: 3469149 (cumulative over frame so far)

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

Assets | Renderbuff... | Framebuffe... | Textures | Frame Ove... | Shaders | Automated ...

Shader	Linked Prog...	Fragments	Total Cycles	Percent Cycles	Cycles	A	L/S	T	Uniform Reg...	Work Reg...
365	366	1,094,147	18,053,426	70.262	16.5	50	12	11	7	8
368	369	93,311	1,679,598	6.537	18	55	12	11	7	8
230	231	465,537	931,074	3.624	2	3	2	2	0	2
200	201	921,600	921,600	3.587	1	2	1	1	0	2
209	210	921,600	921,600	3.587	1	2	1	1	1	2
302	303	921,600	921,600	3.587	1	2	1	0	0	2
218	219	301,236	602,472	2.345	2	3	1	2	1	2
362	363	31,752	555,660	2.163	17.5	55	13	12	7	8

Total Cycles: 25694602 (cumulative over frame so far)

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

Frame buffer resolution 1280 x 720 = 922 K pixels

Num. verts = 272 K vertices

Total vertex cycle count per vertex

3.5M / 272 K = 12.9 cycles/frame/vert

Total fragment cycle count per pixel

25.7M / 922K = 27.9 cycles/frame/pixel

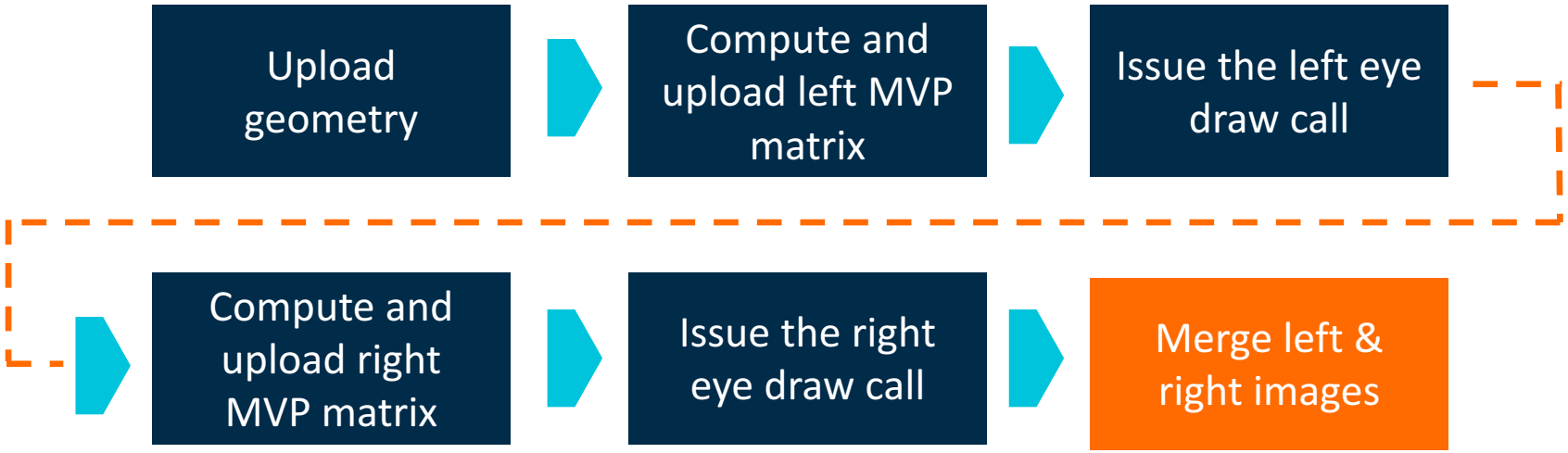
Compare these figures vs your GPU processing budget!

Single-pass stereo rendering vs multi-pass

arm

Multi-pass vs single-pass stereo rendering

Traditional multi-pass pipeline for rendering stereo images



Single-pass (multiview) pipeline for rendering stereo images

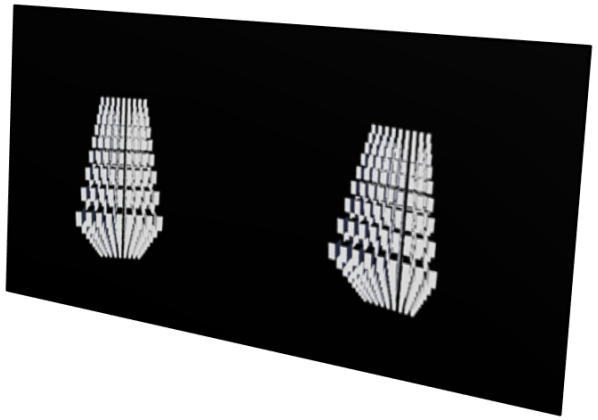
Vertex shader with single-pass (multiview)

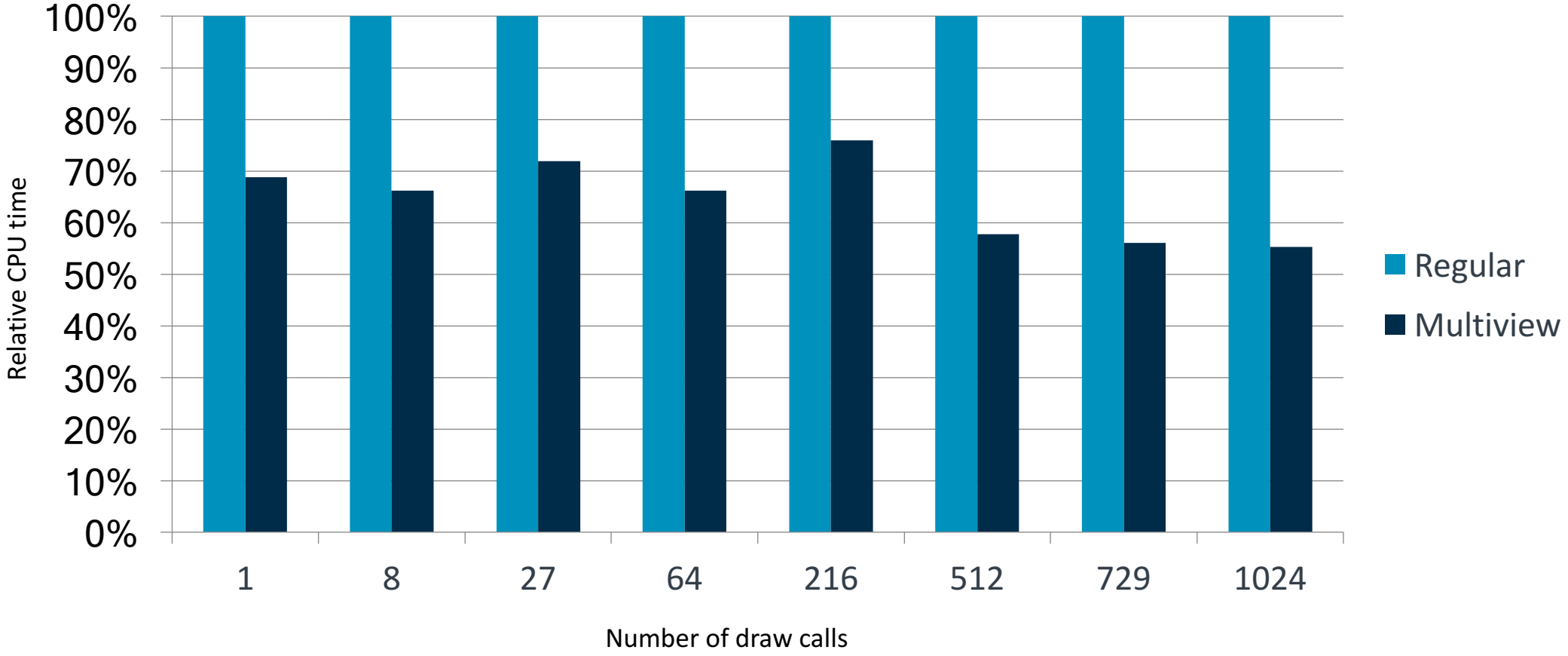
```
#version 300 es
#extension GL_OVR_multiview2 : enable
precision highp float;
layout(num_views = 2) in;
in vec3 vertexPosition;
in vec2 UVCoordinates;
out vec2 texCoord;
uniform mat4 MVP[2];


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

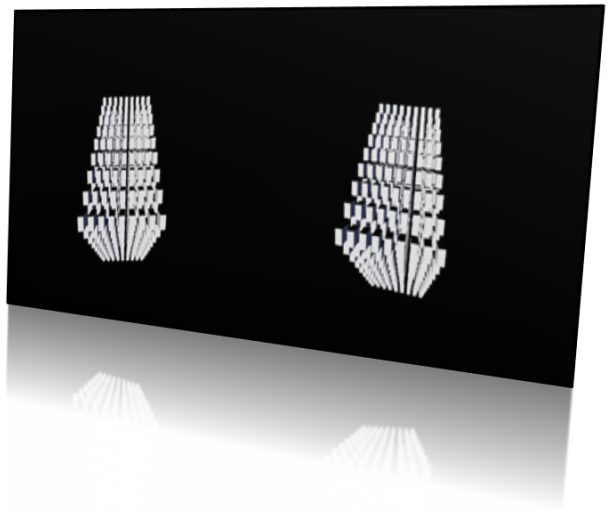
← **This line is executed N view times**

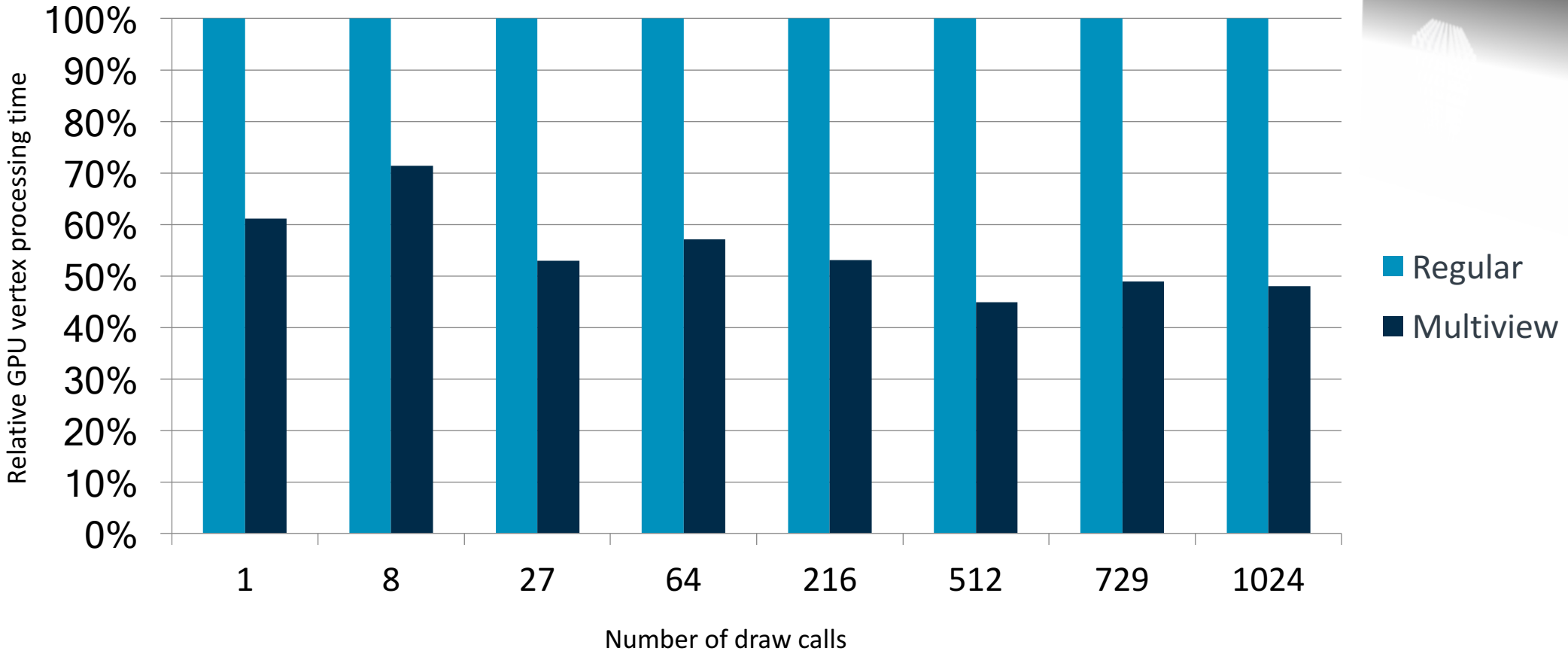
Single-pass (multiview): CPU load





Single pass (multiview): GPU vertex load





Single-pass stereo rendering power efficiency

Single-pass stereo rendering

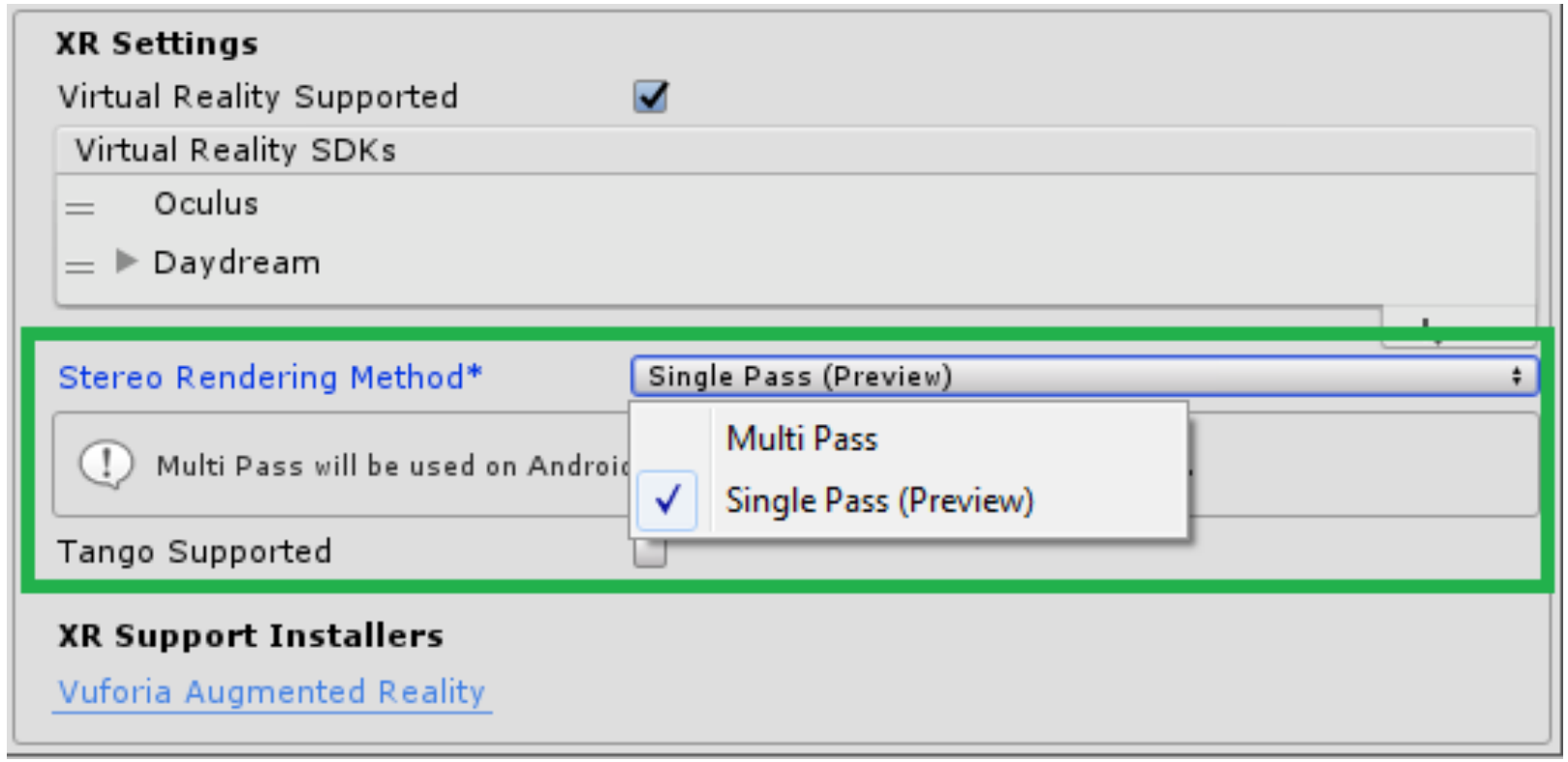
- Halves the number of draw calls
- Reduces the load on the vertex processing
- Reduces the bandwidth usage

Our measurements: ~ 10% reduction of power consumption

- Make the game last longer
- Increase complexity of your scene

arm

Enable single-pass stereo rendering!



Mobile VR best practices

arm

Mobile VR best practice

Enable 4x MSAA

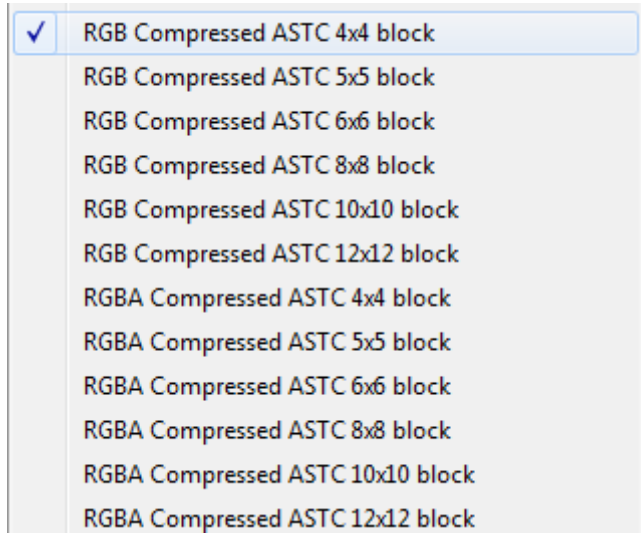
- Virtually for free in ARM Mali GPUs

Try 8x MSAA for best visual quality

- Limited impact in ARM Mali GPUs if poly count under control

Use texture compression

- ASTC provides wide range of choices



Use optimized rendering techniques

- Allow better performance with less use of runtime resources

Rendering techniques based on local cubemaps

Technique	Cubemap	Apply 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

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

* Unity Asset Store
** Blog Post - Combined Reflections: Stereo Reflections in VR
*** Blog post - Refraction Based on Local Cubemaps

Dynamic soft shadows based on local cubemaps

Video - Reflections and shadows based on local cubemaps

Reflections based on local cubemap

Video - Reflections and shadows based on local cubemaps

New features coming to VR

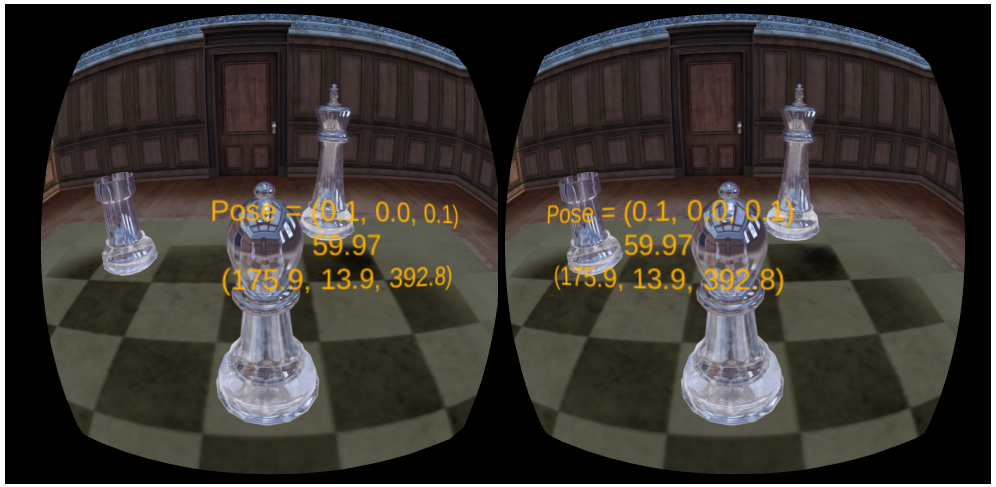
Vulkan coming to VR

- Same benefits as in non-VR apps:
  - Lower load on the CPU and more efficient use of multi-core architecture
  - Less bandwidth use

Inside-out mobile VR tracking already a reality

- Using Google ARCore SDK on Samsung Galaxy S8

  Arm Mali G71 GPU
- Using Google Tango SDK on ASUS AR Zenfone
- New type of VR games and experiences
- [Read more about this on Unity's blog]




LILA'S TALE
arm

Lila’s Tale Trailer

Lila's Tale Trailer on YouTube

Game planning

GPU BUDGET

arm

What we wanted?

A really beautiful game

What we wanted?

Simple to play without hands (tap)

What we wanted?

Scenario with a lot of colors

What we wanted?

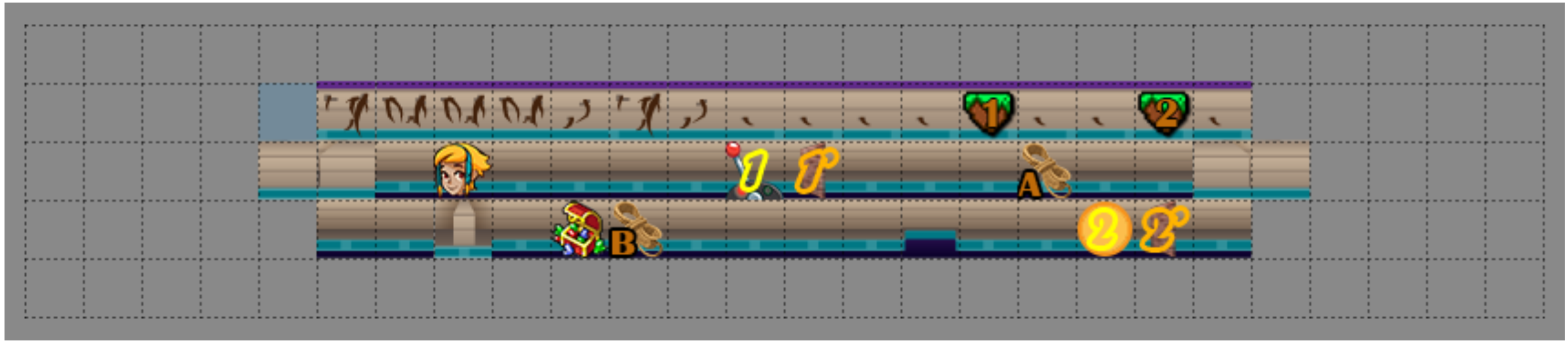
Smooth run

OPTIMIZATION STARTS ON DAY 1

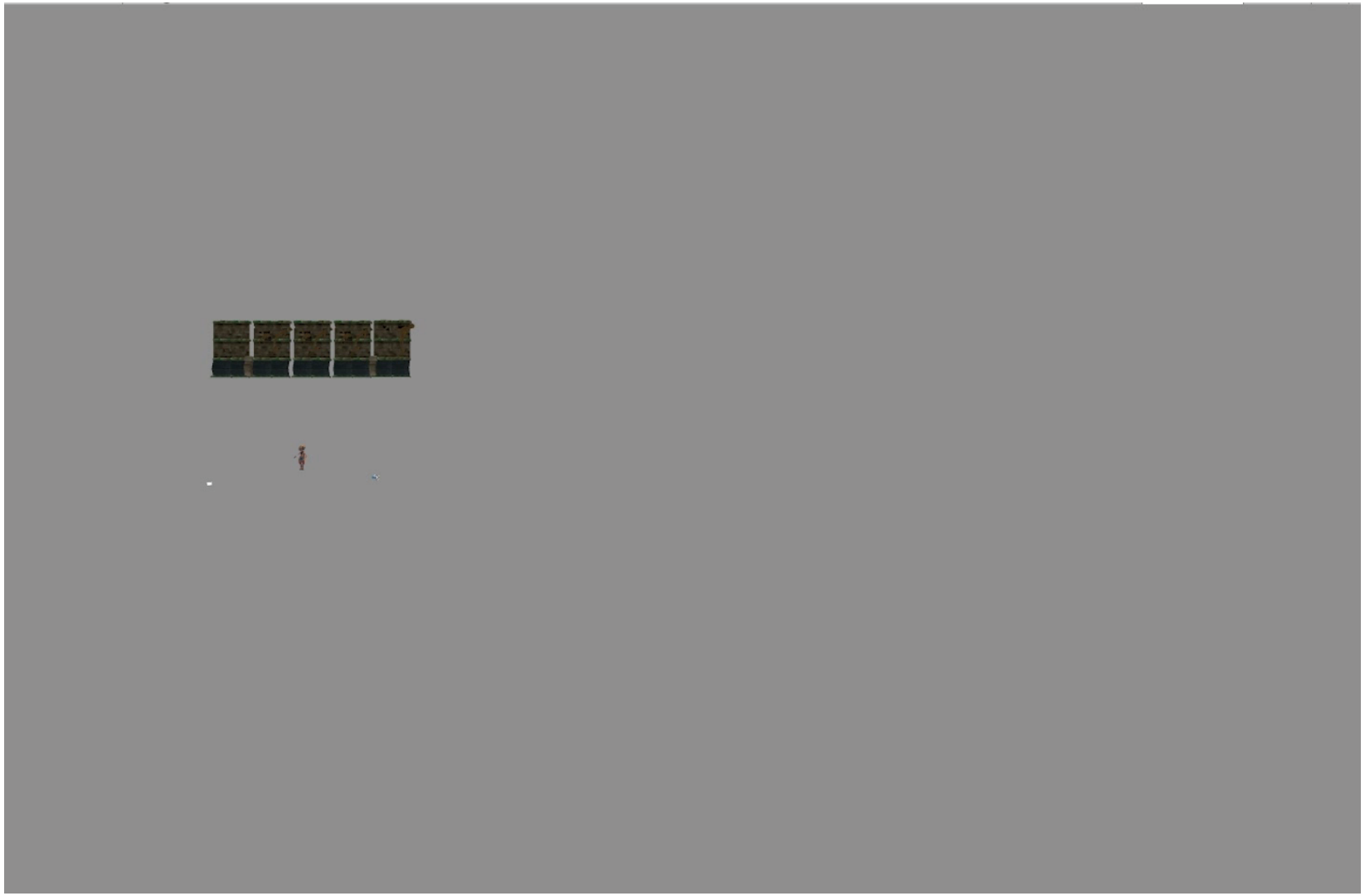
Level editor

- Iterate faster
- No developer dependence
- Make GD easier

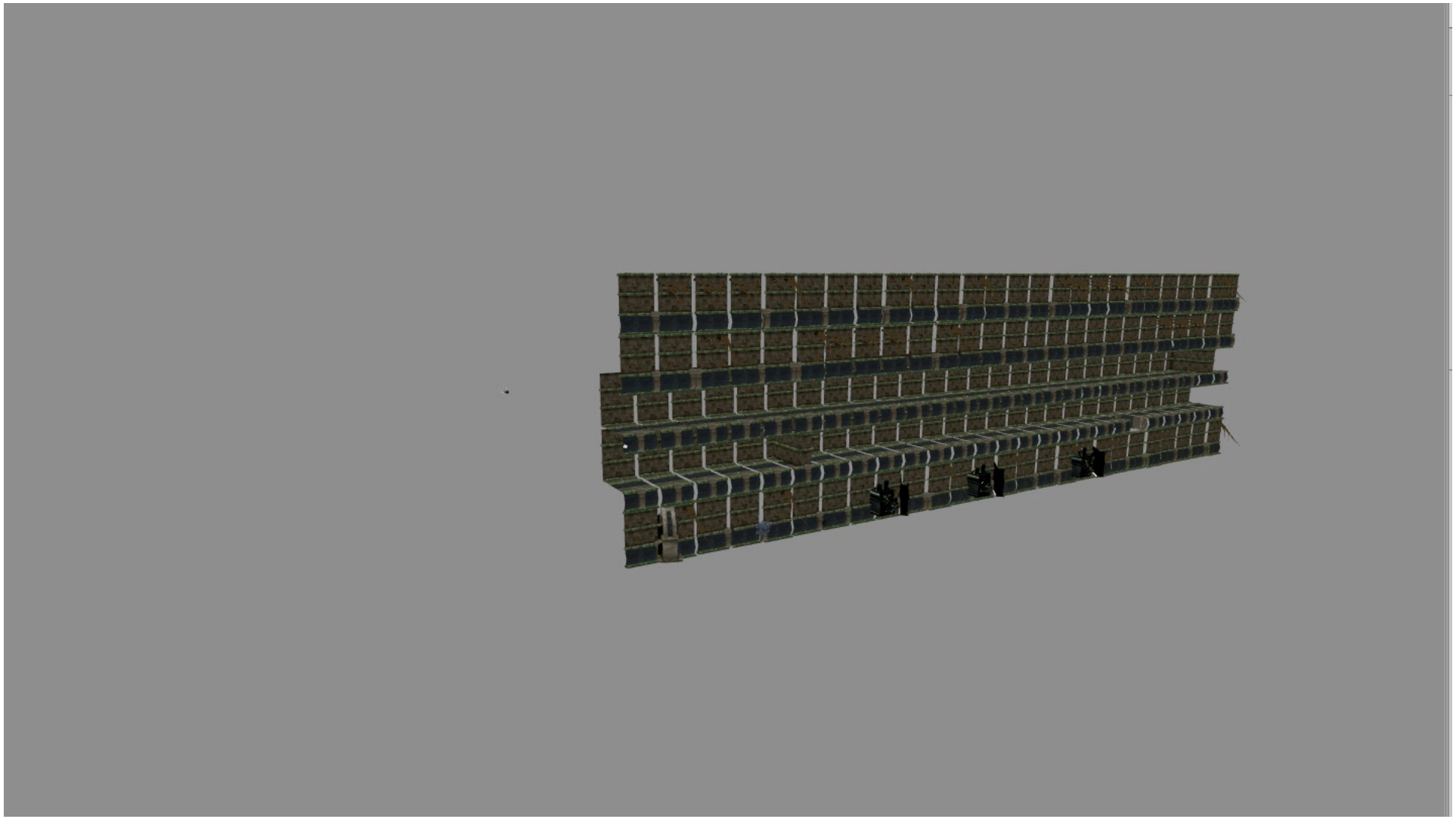
Level editor - 2D Design



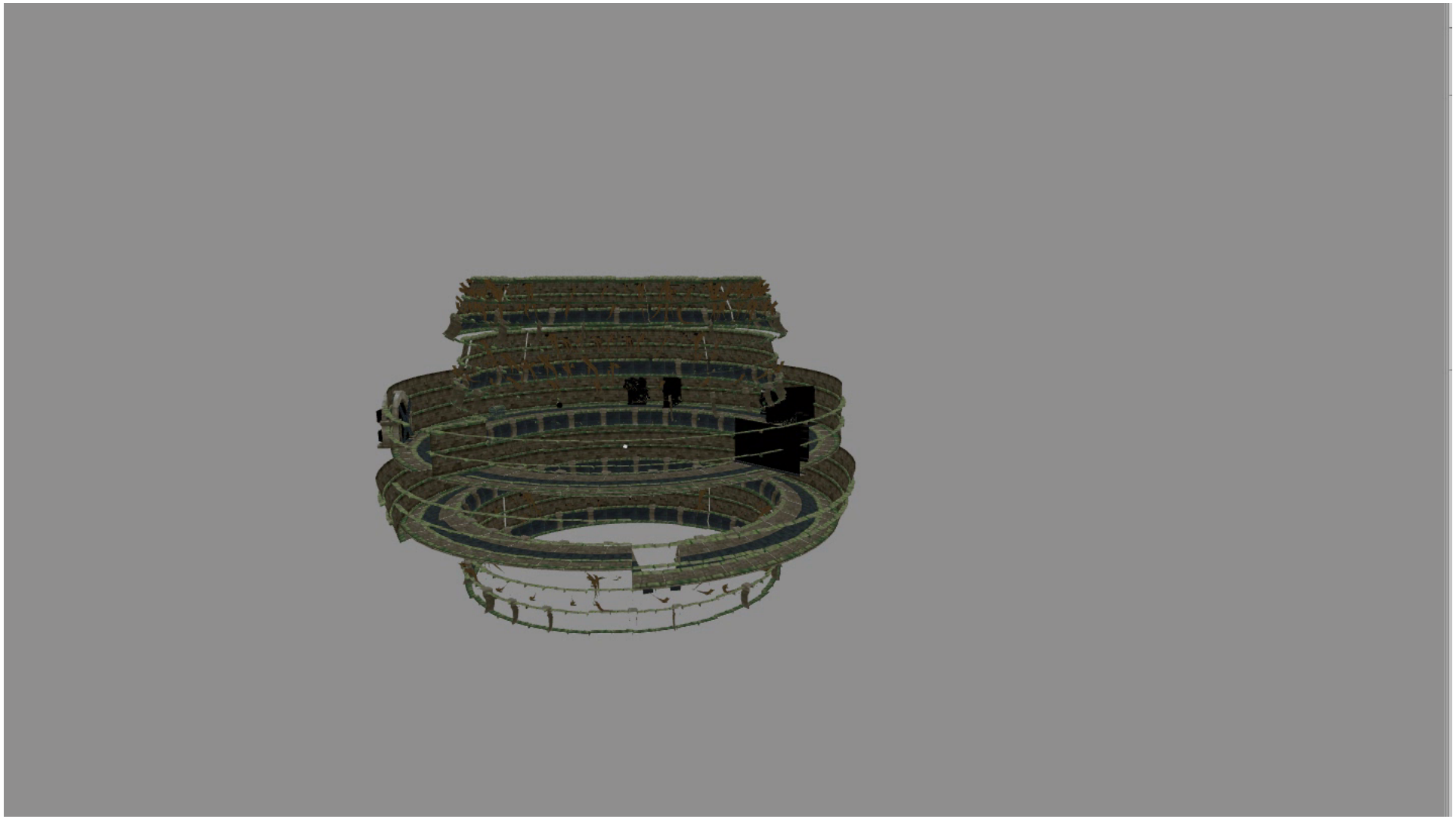
Level editor - 3D Tile Creation



Level editor - Dungeon Creation



Level editor - Adjust Heights

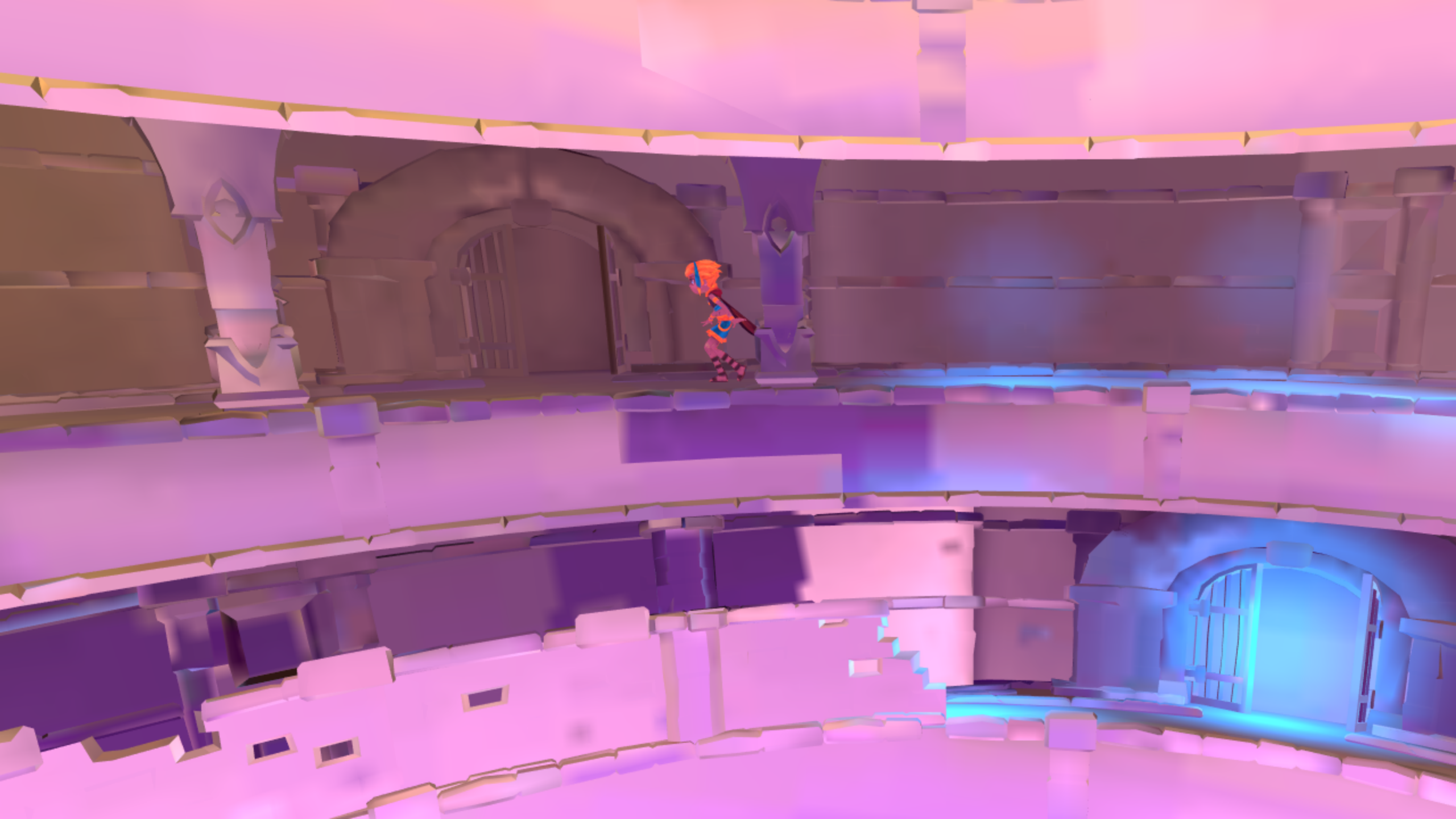


Level Editor

Reduce draw calls

Level Editor

Generate Lightmaps



Game optimization

MALI GRAPHICS DEBUGGER

arm

Lila's Tale GPU processing budget

Samsung Galaxy S7: octa-core Arm CPU and Arm Mali-T880 MP12 GPU

- 12 GPU cores @ 650 MHz – 12 x 650 M single cycle operations per second
- Target FPS: 60
- Res = *2,560 x 1,440 = 3.68 M pixels*
- *Num. Verts = 671,436*

fragCycleBudget = (130 M cycles/frame) / (3.68 M pixels) ~ 35 cycles/frame/pixel

vertCycleBudget = (130 M cycles/frame) / (0.67 M verts) ~ 194 cycles/frame/vertex

Lila's Tale fragment and vertex cycle count from MGD

Shader	Linked Progra...	Vertices	Total Cycles	Percent Cycles	Cycles	A	L/S	T	Uniform Reg...	Work Reg...
319	321	482,322	4,340,898	59.621	9	17	9	0	14	3
322	324	70,605	1,200,285	16.485	17	44	17	0	18	6
325	327	34,545	621,810	8.54	18	43	18	0	16	8
331	333	37,704	527,856	7.25	14	28	14	0	16	4
334	336	34,545	518,175	7.117	15	26	15	0	14	6
307	309	10,857	65,142	0.895	6	14	6	0	11	3
238	240	291	2,328	0.032	8	16	8	0	13	3
268	270	216	1,944	0.027	9	16	9	0	13	3
340	342	138	966	0.013	7	11				
328	330	36	504	0.007	14	30				
337	339	60	480	0.007	8	16				
316	318	108	432	0.006	4	12				
187	189	6	42	0.001	7	12				
199	201	3	12	0	4	5				

Total Cycles: 7280874 (cumulative over frame so far)

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

Total Cycles: 7280874 (cumulative over frame so far)

Frame buffer resolution = 2,560 x 1,440 = 3.68 M pixels

Num. verts = 671,436

Total vertex cycle count/frame/vert

7.28 M cycles/frame /0.67 M verts ~ 11 cycles/frame/vert

11 cycles/frame/vert << 194 cycles/frame/vert

arm

Lila's Tale fragment and vertex cycle count from MGD

Shader	Linked Progra...	Fragments	Total Cycles	Percent Cycles	Cycles	A	L/S	T	Uniform Reg...	Work Reg...
320	321	5,754,737	17,264,212	58.587	3	9	3	3	2	4
200	201	3,686,400	3,686,400	12.51	1	2	1	1	0	2
317	318	3,686,400	3,686,400	12.51	1	4	1	0	1	2
239	240	309,934	619,868	2.104	2	7	2	2	1	3
308	309	303,008	696,919	2.365	2.3	8	2	1	3	3
326	327	239,419	2,394,190	8.125	10	22	7	10	8	4
335	336	92,409	924,090	3.136	10	21	5	10	11	4
341	342	23,413	117,065	0.397	5	10	1	5	1	2
323	324	7,284	65,556	0.222	9	19	6	9	5	3
338	339	4,526	9,052	0.031	2	4	2	1	1	2
188	189	1,955	1,955	0.007	1	3	1	1	0	2
329	330	220	2,178	0.007	9.9	31	4	4	6	3
2	3	0	0	0	1					

Total Cycles: 29467884 (cumulative over frame so far)

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

Total Cycles: 29467884 (cumulative over frame so far)

Total fragment cycle count/frame/pixel

29.5M cycles/frame / 3.68 M pixels ~ 8 cycles/frame/pixel

8 cycles/frame/pixel < 35 cycles/frame/pixel

Single-pass vs Multi-pass

arm

Multi-pass vs single-pass (Lila's Tale, MGD stats)

Multi-pass

1,434,896 vertices, 163 draw calls, 159 instances, 1,434896 indices …

- Process 0 (com.skullfishstudios.lilastaleARMmultipassDEV)
  - Frame 0 6 vertices, 11 draws, 1 instance, 0 instanced vertices, 6 indices, 0 unique indices, 11 render passes
  - Frame 1 6 vertices, 5 draws, 1 instance, 0 instanced vertices, 6 indices, 0 unique indices, 4 render passes
  - Frame 2 1434062 vertices, 163 draws, 159 instances, 0 instanced vertices, 1434062 indices, 667934 unique indices, 4 render passes
  - Frame 3 1434896 vertices, 163 draws, 159 instances, 0 instanced vertices, 1434896 indices, 668386 unique indices, 4 render passes
  - Frame 4 1434896 vertices, 163 draws, 159 instances, 0 instanced vertices, 1434896 indices, 668386 unique indices, 4 render passes
  - Frame 5 1434896 vertices, 163 draws, 159 instances, 0 instanced vertices, 1434896 indices, 668386 unique indices, 4 render passes

Multi-pass vs single-pass (Lila's Tale, MGD stats)

Single-pass

729,356 vertices, 85 draw calls, 83 instances, 729,356 indices …

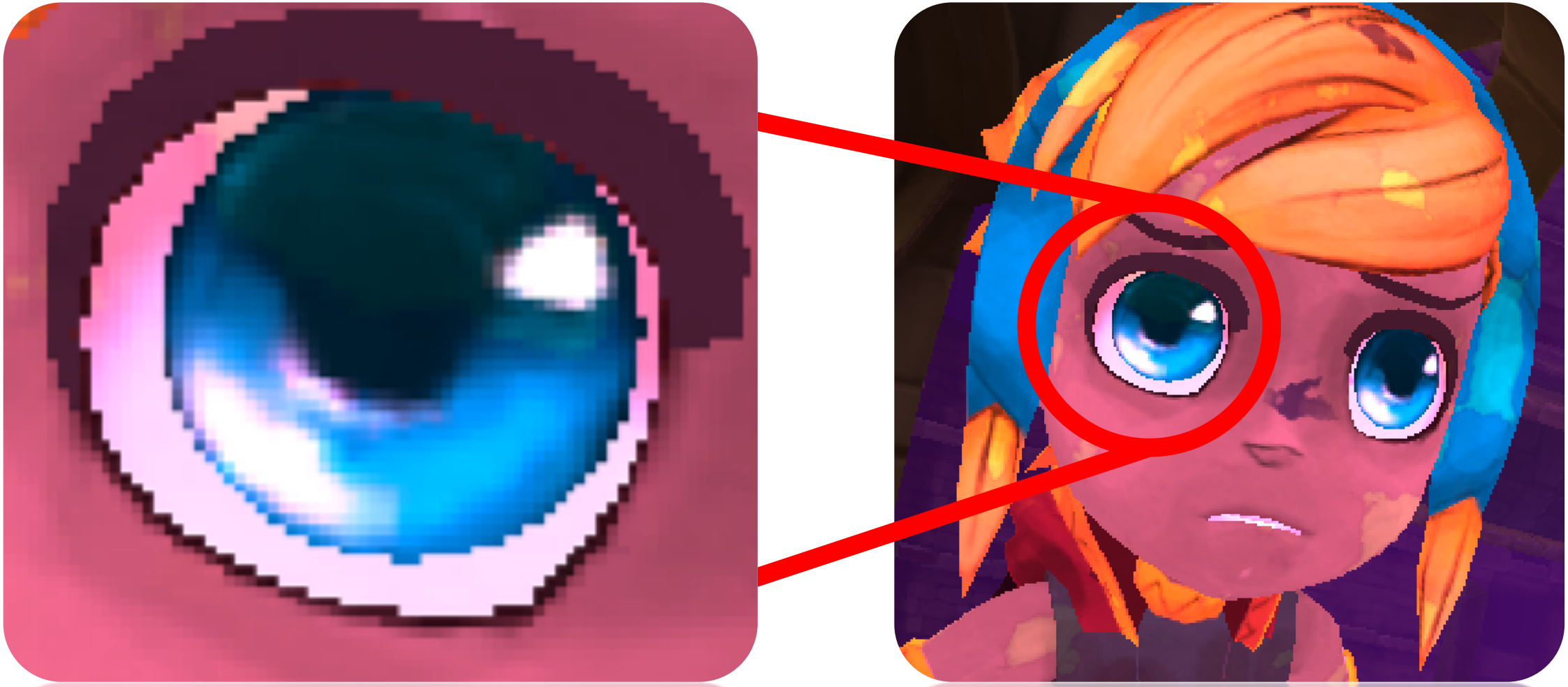
- Process 0 (com.skullfishstudios.lilastaleARMsingleipass)
  - Frame 0 12 vertices, 7 draws, 2 instances, 0 instanced vertices, 12 indices, 4 unique indices, 7 render passes
  - Frame 1 12 vertices, 4 draws, 2 instances, 0 instanced vertices, 12 indices, 4 unique indices, 3 render passes
  - Frame 2 6 vertices, 3 draws, 1 instance, 0 instanced vertices, 6 indices, 0 unique indices, 3 render passes
  - Frame 3 6 vertices, 3 draws, 1 instance, 0 instanced vertices, 6 indices, 0 unique indices, 3 render passes
  - Frame 4 729356 vertices, 85 draws, 83 instances, 0 instanced vertices, 729356 indices, 346289 unique indices, 3 render passes
  - Frame 5 729356 vertices, 85 draws, 83 instances, 0 instanced vertices, 729356 indices, 346289 unique indices, 3 render passes

MSAA & ASTC

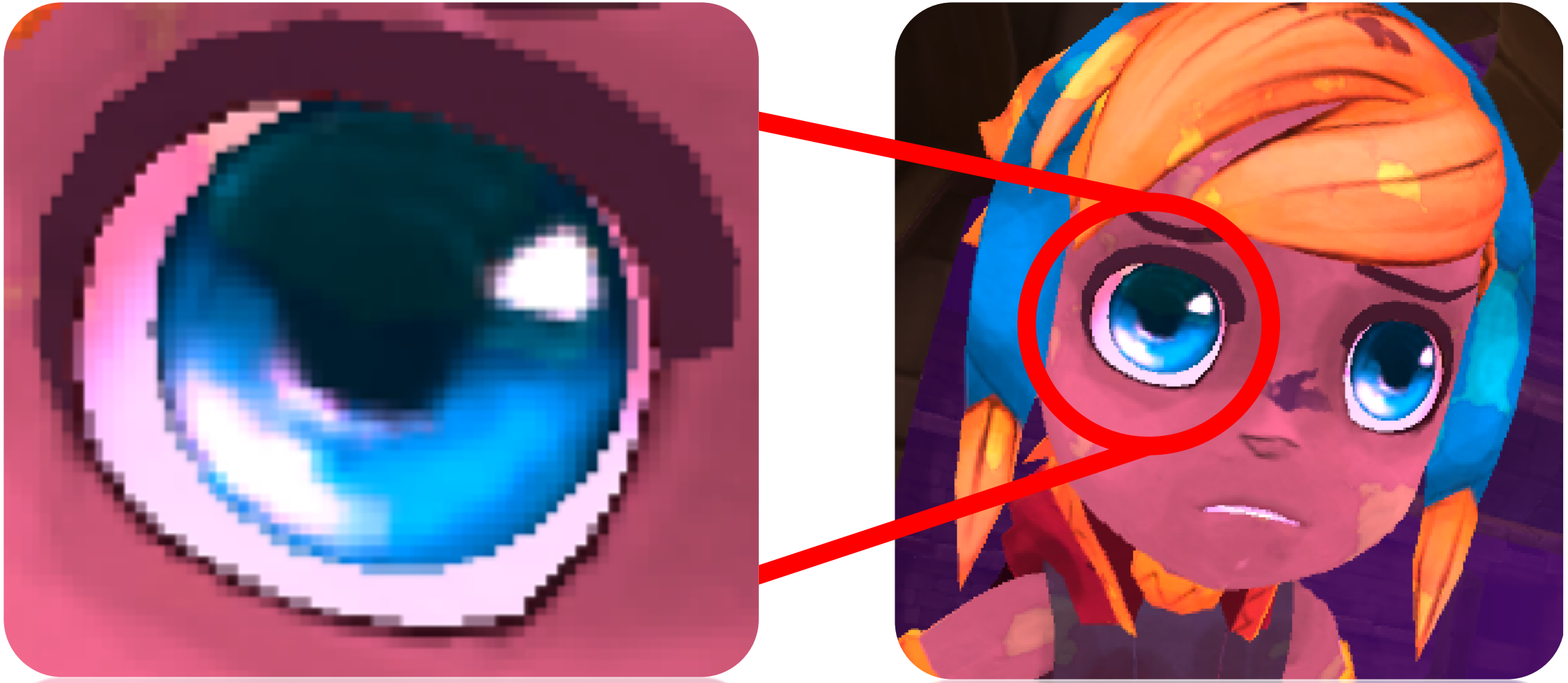
arm

MSAA

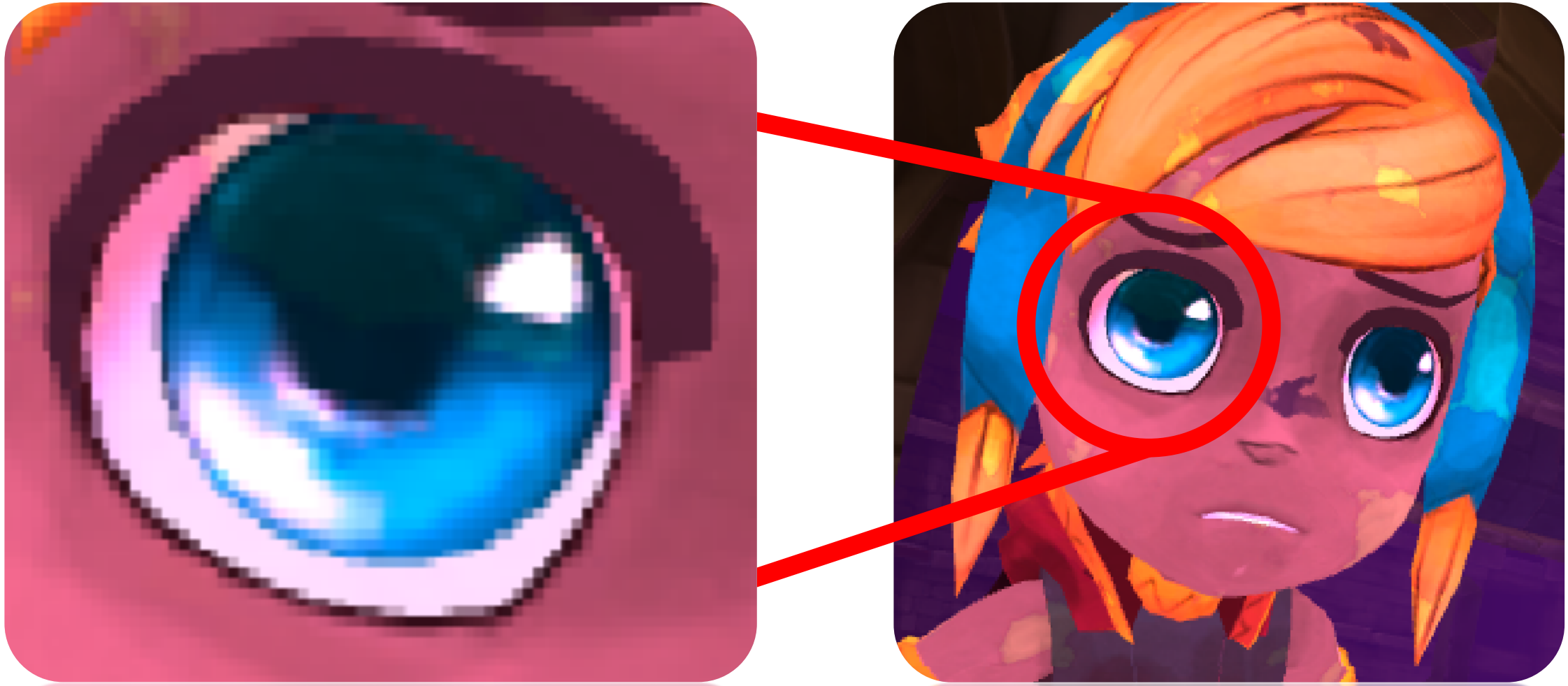
MSAA 4x is almost **for free** in ARM GPU



MSAA DISABLE



MSAA 2X



MSAA 4X

Wrap up

arm

Wrap up

Plan your game using the GPU processing budget

Update to Unity 5.6/2017 to benefit from:

- MGD integration
- Single-pass stereo rendering
- Native support for Daydream and Google Cardboard

Improve VR performance, quality and bandwidth by using:

- 4x MSAA (try 8x MSAA for superior quality)
- ASTC
- Rendering techniques based on local cubemaps

Big new coming features to mobile VR

- Vulkan
- Inside-out mobile VR tracking

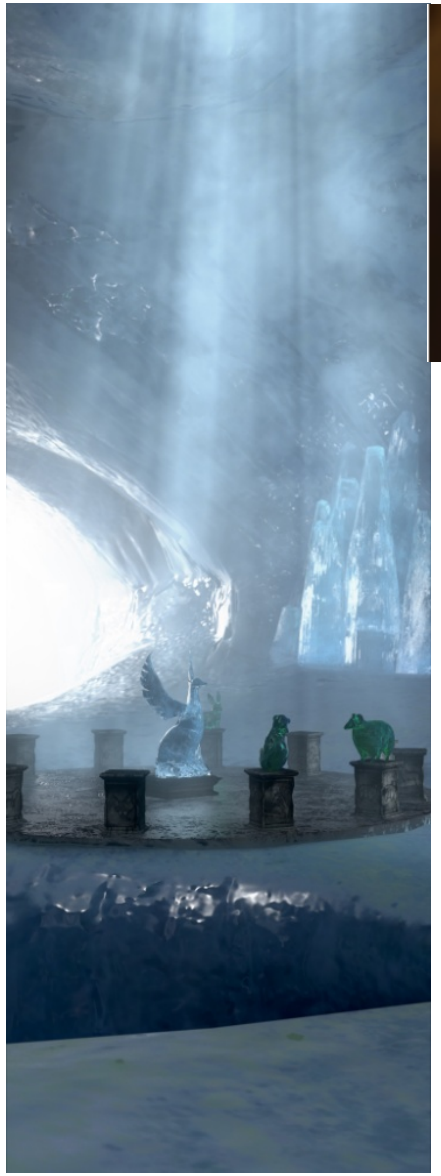
arm

Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!
감사합니다
धन्यवाद

arm

arm

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
  - https://developer.arm.com/graphics/unity/unity-guides-and-white-papers
  - 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.assetstore.unity3d.com/en/#!/content/61640

arm