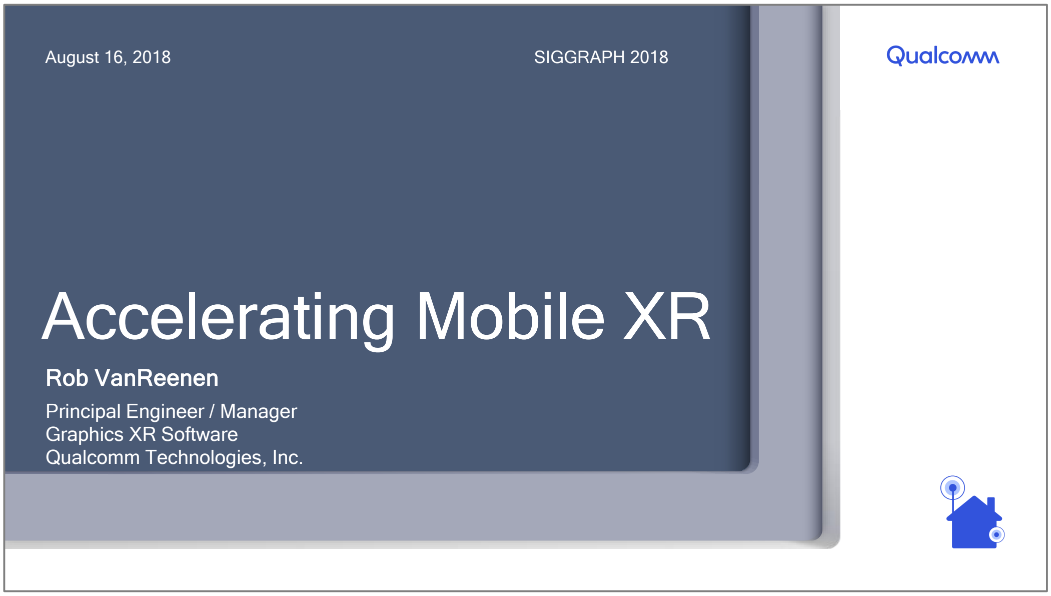


August 16, 2018

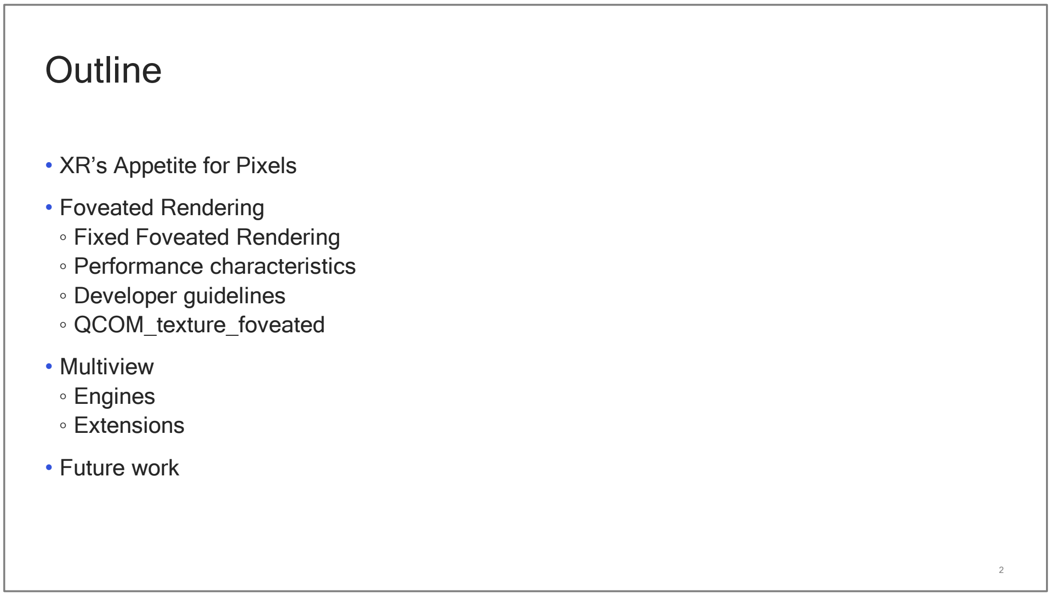
SIGGRAPH 2018

Qualcomm

Accelerating Mobile XR

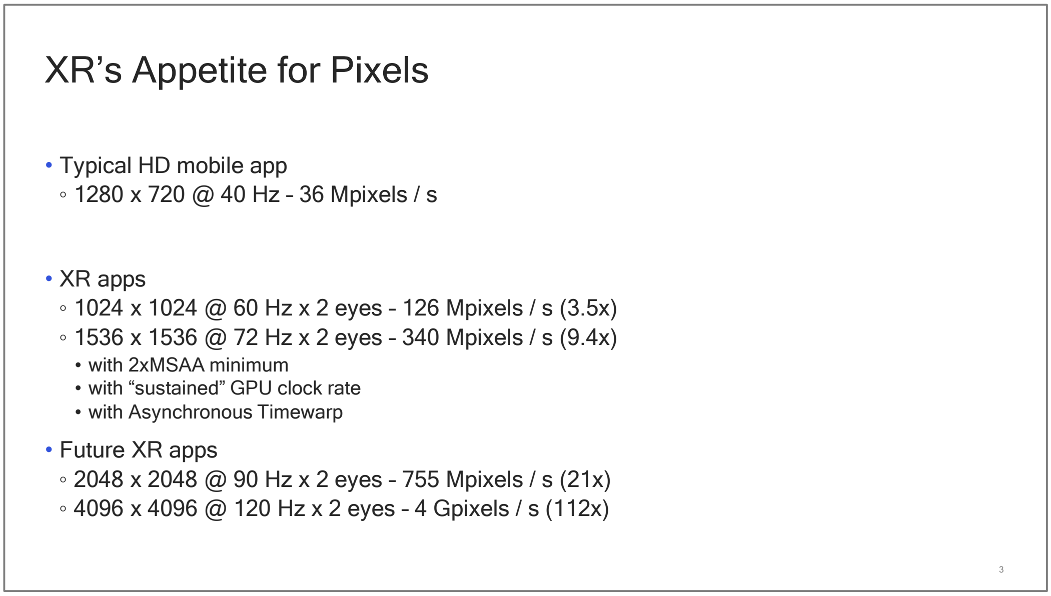
Rob VanReenen

Principal Engineer / Manager
Graphics XR Software
Qualcomm Technologies, Inc.



Outline

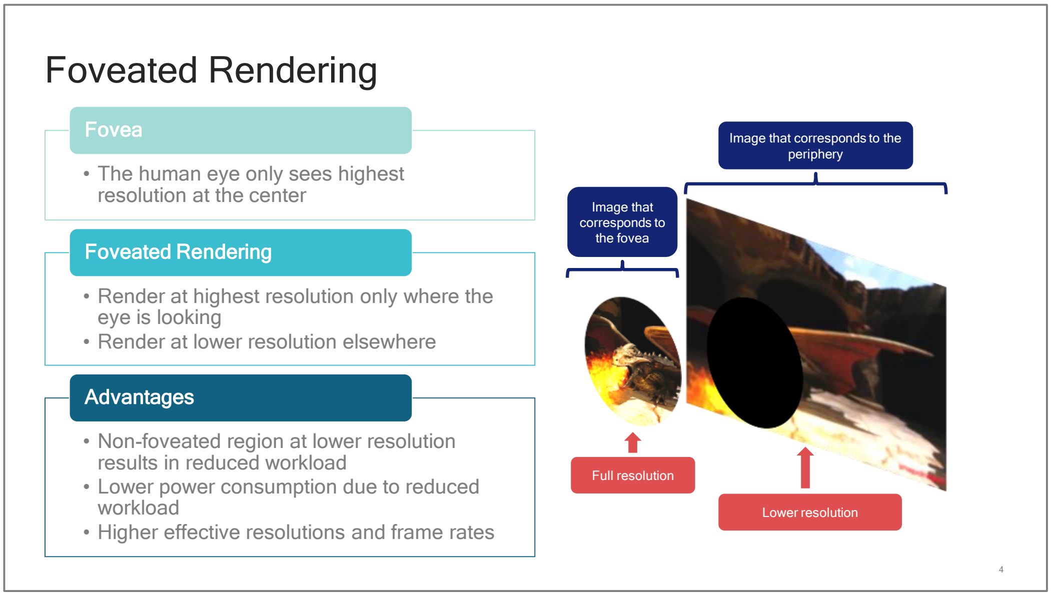
- XR’s Appetite for Pixels
- Foveated Rendering
  - Fixed Foveated Rendering
  - Performance characteristics
  - Developer guidelines
  - QCOM_texture_foveated
- Multiview
  - Engines
  - Extensions
- Future work



XR’s Appetite for Pixels

- Typical HD mobile app
  - 1280 x 720 @ 40 Hz - 36 Mpixels / s
- XR apps
  - 1024 x 1024 @ 60 Hz x 2 eyes - 126 Mpixels / s (3.5x)
  - 1536 x 1536 @ 72 Hz x 2 eyes - 340 Mpixels / s (9.4x)
    - with 2xMSAA minimum
    - with “sustained” GPU clock rate
    - with Asynchronous Timewarp
- Future XR apps
  - 2048 x 2048 @ 90 Hz x 2 eyes - 755 Mpixels / s (21x)
  - 4096 x 4096 @ 120 Hz x 2 eyes - 4 Gpixels / s (112x)

- 720p
- Timewarp is preempting twice a frame for a couple ms
- Oculus Go now running some content @ 72 Hz
- XR demands that we push a lot more pixels



Foveated Rendering

Fovea

- The human eye only sees highest resolution at the center

Foveated Rendering

- Render at highest resolution only where the eye is looking
- Render at lower resolution elsewhere

Advantages

- Non-foveated region at lower resolution results in reduced workload
- Lower power consumption due to reduced workload
- Higher effective resolutions and frame rates

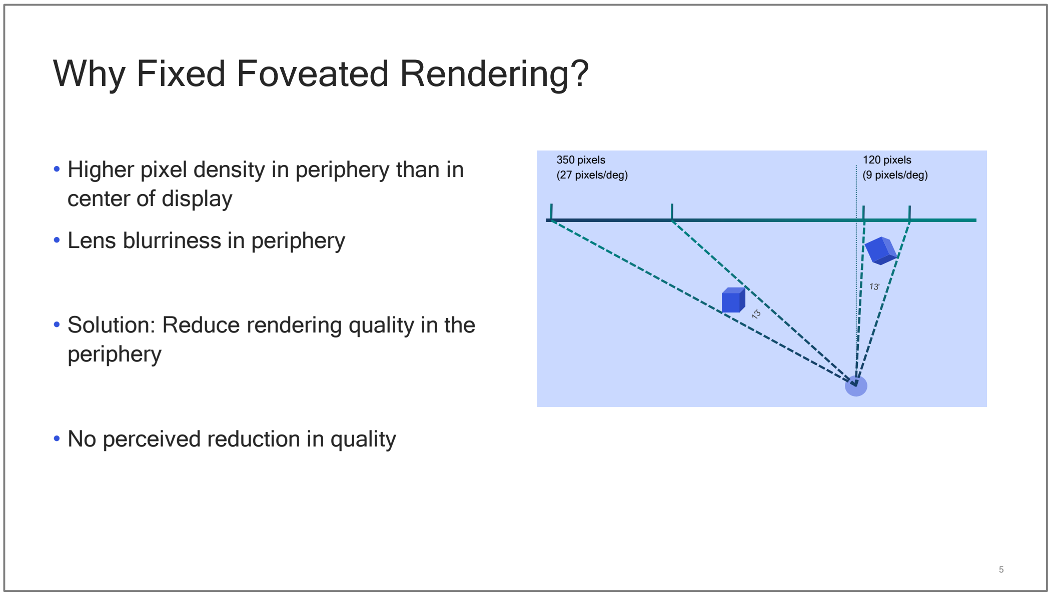
Image that corresponds to the periphery

Image that corresponds to the fovea

Full resolution

Lower resolution

- The fovea accounts for a small fraction of our field of view
- About 5 out of a total of 210 degrees in human stereoscopic vision
- Rule of thumb
- Approx. half of the nerve fibers in the optic nerve carry information from the fovea



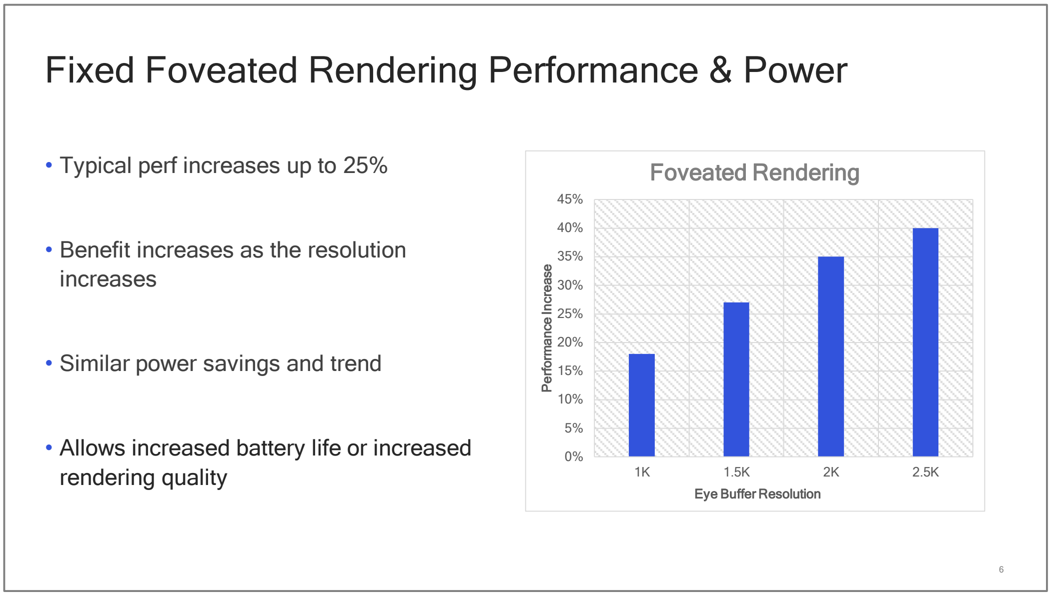
Why Fixed Foveated Rendering?

- Higher pixel density in periphery than in center of display
- Lens blurriness in periphery
- Solution: Reduce rendering quality in the periphery
- No perceived reduction in quality

- Foveated rendering is not just for use with eye-tracking

- Projecting the virtual world onto a flat display produces a greater pixel density in the periphery than in the center
- As FOV approaches 180 degrees, pixels per degree approaches infinity
- Lenses are blurrier in the periphery

- Fixed foveation corrects this pixel density disparity by reducing quality outside of the center of the eye buffer
- Can’t be as aggressive as with eye-tracking
- Rendering workload reduction with no perceived reduction in quality!
- Free perf!

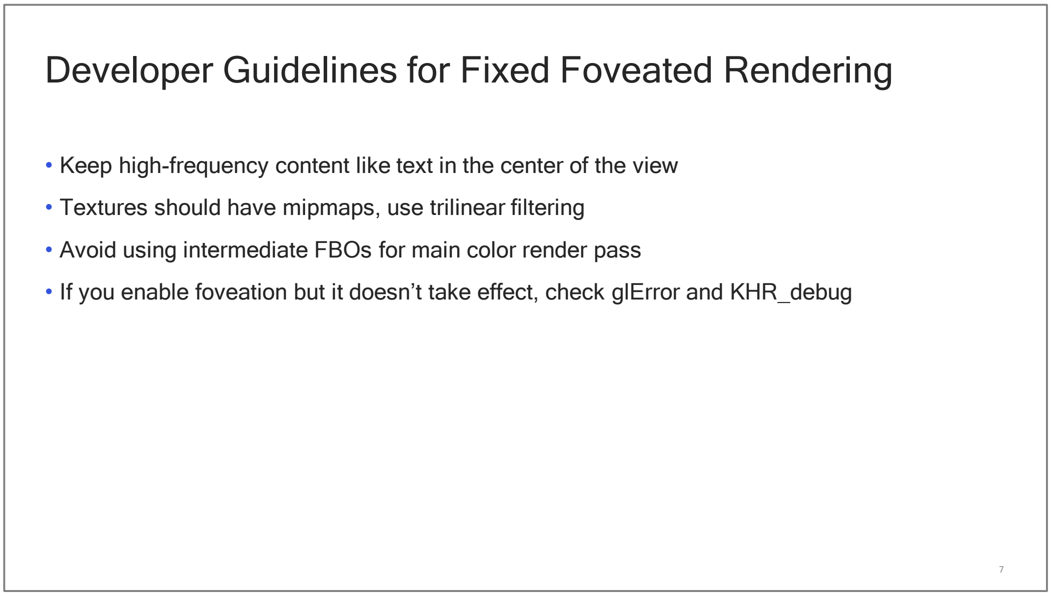


Fixed Foveated Rendering Performance & Power

- Typical perf increases up to 25%
- Benefit increases as the resolution increases
- Similar power savings and trend
- Allows increased battery life or increased rendering quality

Oculus: “FFR can result in a 25% gain in performance with pixel-intensive applications.”

Used by a significant number of Oculus Go apps



Developer Guidelines for Fixed Foveated Rendering

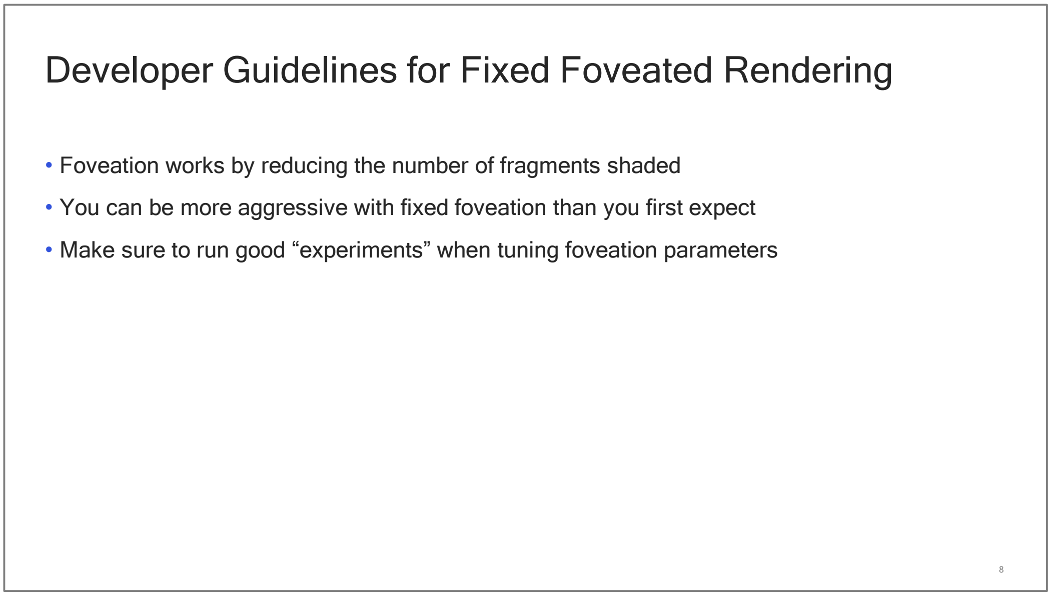
- Keep high-frequency content like text in the center of the view
- Textures should have mipmaps, use trilinear filtering
- Avoid using intermediate FBOs for main color render pass
- If you enable foveation but it doesn't take effect, check glError and KHR_debug

Keep high-frequency content like text in the center of the view
- Or use a separate text layer in VR composition/warp

Textures should have mipmaps, use trilinear filtering and maybe aniso

Avoid using intermediate FBOs for main color render pass
- Main render pass must enable foveated rendering to see perf/power gains
- Current VR SDKs only allow foveated rendering on the main swap chain
- If you must, enable foveated rendering "by hand" on the intermediate FBO

If you enable foveation but it doesn't take effect, check glError and KHR_debug messages
- Are you using tessellation, geometry shaders, or compute shaders?
- It's a good general rule to avoid using these for XR content.



Developer Guidelines for Fixed Foveated Rendering

- Foveation works by reducing the number of fragments shaded
- You can be more aggressive with fixed foveation than you first expect
- Make sure to run good “experiments” when tuning foveation parameters

Foveation works by reducing the number of fragments shaded

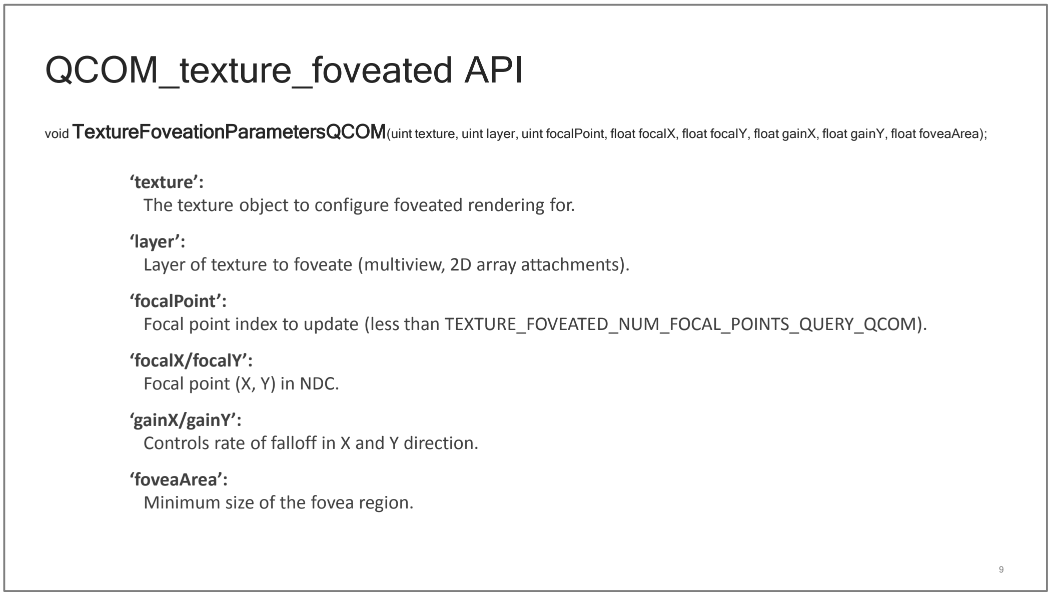
- More complex fragment shaders = more perf/power gains
- Higher resolutions = more perf/power gains
- If you have extremely simple fragment shaders, there will be limited benefit

You can be more aggressive with fixed foveation than you first expect

- Try multiple settings to find which hits the sweet spot for quality-perf-power
- Always check quality in a headset!

Make sure to run good “experiments” when tuning foveation parameters

- If you ask a “test subject” to find something, they will usually find it – whether it is there or not
- This can be a good test for the quality of your content – you’ll find all sorts of aliasing, blurriness, and flickering that you never knew were there
- Bifocals
- Generally starting with foveation disabled is best to set a baseline
- This is a strange kind of test, because success is hearing, “I can’t see it”



QCOM_texture_foveated API

void TextureFoveationParametersQCOM(uint texture, uint layer, uint focalPoint, float focalX, float focalY, float gainX, float gainY, float foveaArea);

‘texture’:
The texture object to configure foveated rendering for.

‘layer’:
Layer of texture to foveate (multiview, 2D array attachments).

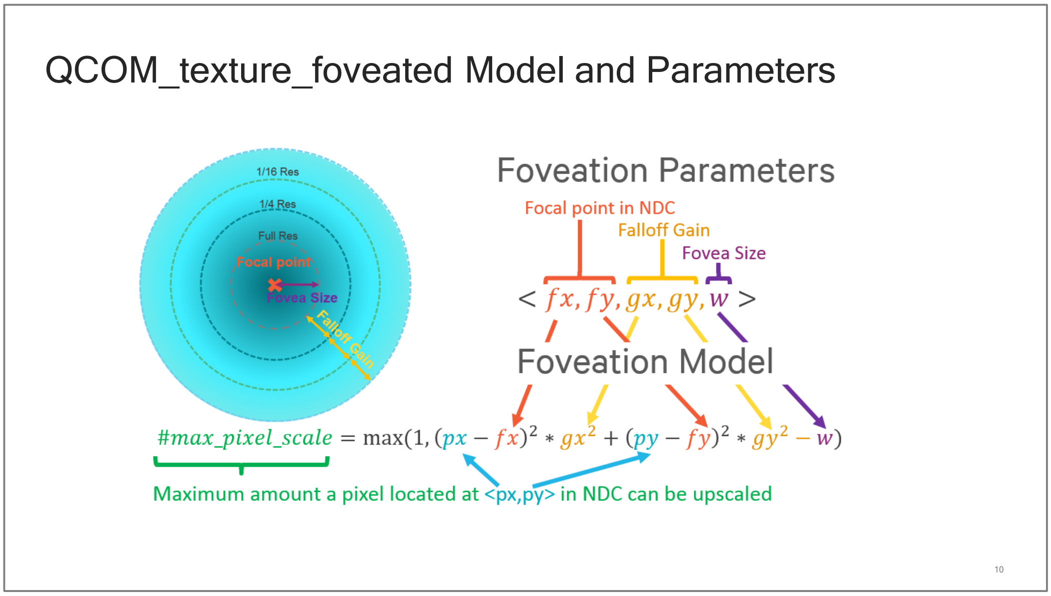
‘focalPoint’:
Focal point index to update (less than TEXTURE_FOVEATED_NUM_FOCAL_POINTS_QUERY_QCOM).

‘focalX/focalY’:
Focal point (X, Y) in NDC.

‘gainX/gainY’:
Controls rate of falloff in X and Y direction.

‘foveaArea’:
Minimum size of the fovea region.

- QCOM_texture_foveated was introduced with Qualcomm® Snapdragon™ 835
- It is enabled by the Snapdragon VR SDK for use with Unity, Unreal, and native applications
- Currently being used pretty extensively on Oculus Go



QCOM_texture_foveated Model and Parameters

Foveation Parameters

Focal point in NDC

Falloff Gain

Fovea Size

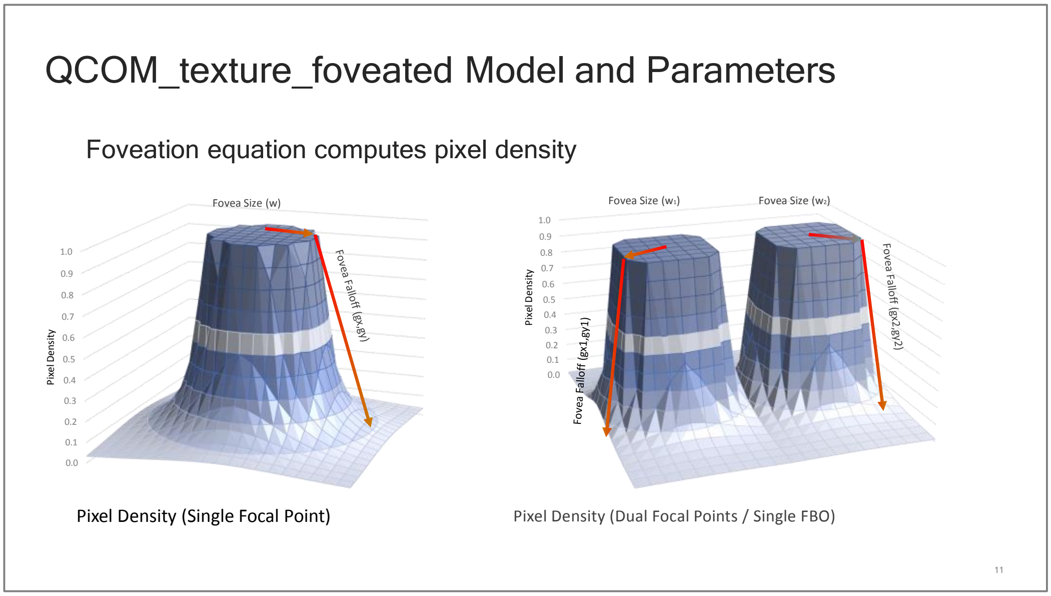
$< fx, fy, gx, gy, w >$

Foveation Model

$$\#max\_pixel\_scale = \max(1, (px - fx)^2 * gx^2 + (py - fy)^2 * gy^2 - w)$$

Maximum amount a pixel located at <px,py> in NDC can be upscaled

- Allows an application to bound the minimum quality of each tile while giving the implementation flexibility on the size and shape of each tile.

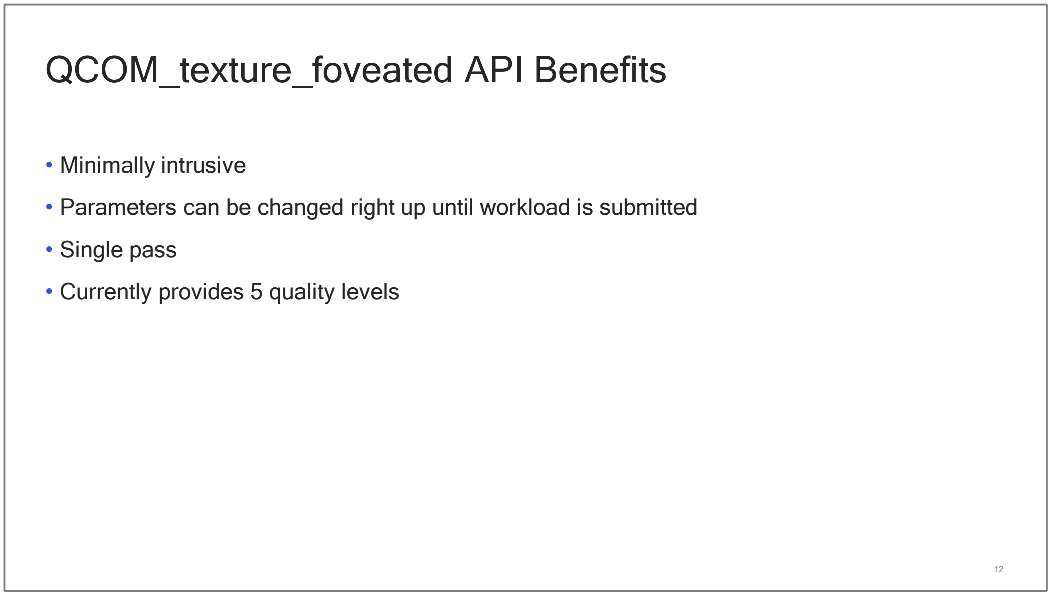


QCOM_texture_foveated Model and Parameters

Foveation equation computes pixel density

Pixel Density (Single Focal Point)

Pixel Density (Dual Focal Points / Single FBO)



QCOM_texture_foveated API Benefits

- Minimally intrusive
- Parameters can be changed right up until workload is submitted
- Single pass
- Currently provides 5 quality levels

12

Minimally intrusive

- Single configurable function call
- Works with single or multiple focal points
- Works with multiview, one eye per texture, and two eyes in one texture
- Does not require rewriting application rendering pipeline or shaders

Parameters can be changed right up until workload is submitted

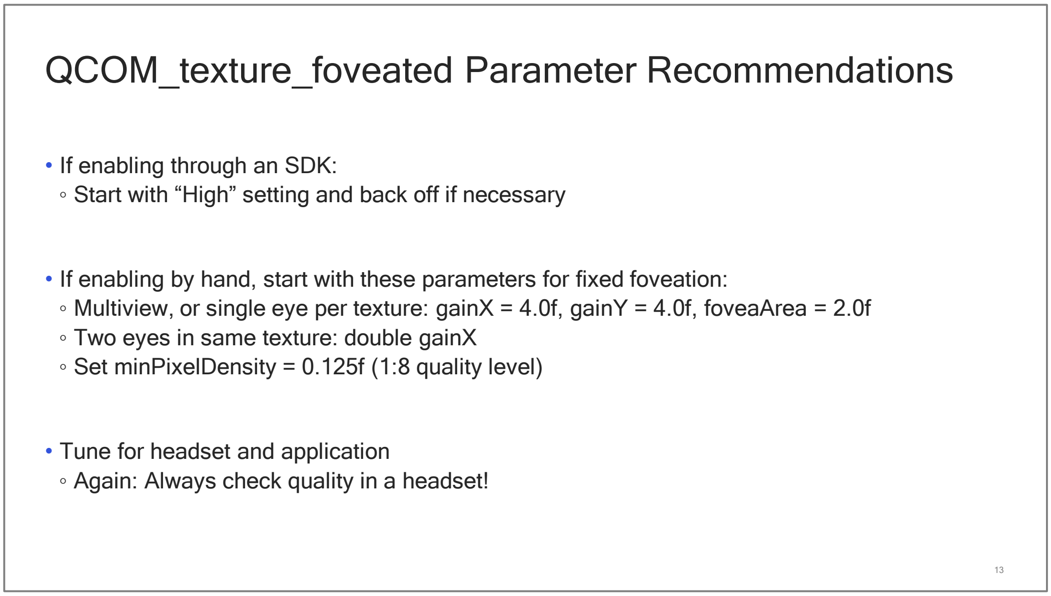
- Enable/disable foveation, adjust falloff, area, or eye focal point

Single pass

- Avoids executing the drawcalls multiple times
- Avoids extra memory read/write overhead

Currently provides 5 quality levels

- 1:1, 1:2, 1:4, 1:8, 1:16

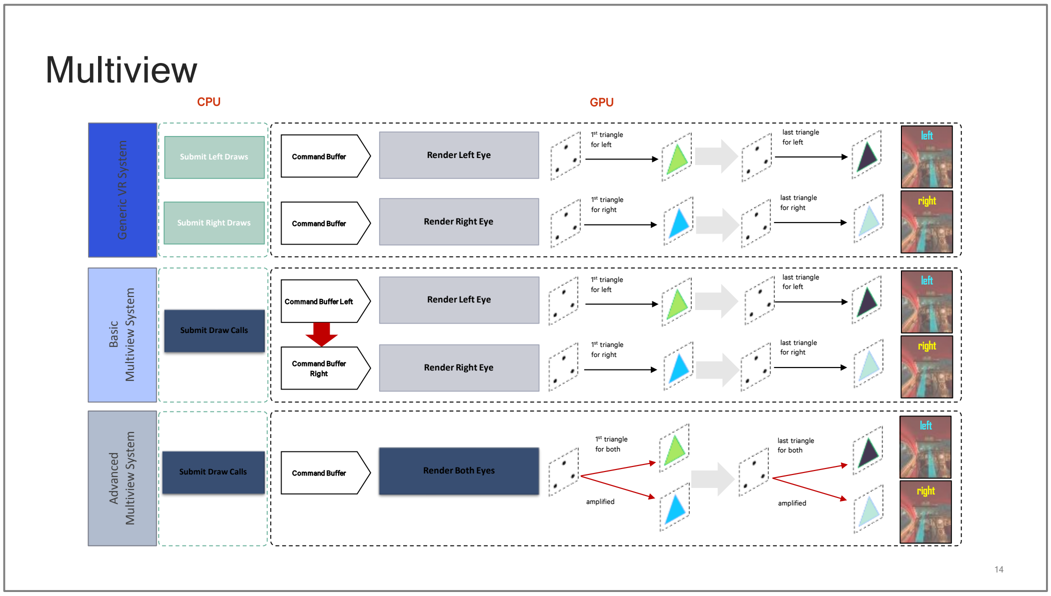


QCOM_texture_foveated Parameter Recommendations

- If enabling through an SDK:
  - Start with “High” setting and back off if necessary

- If enabling by hand, start with these parameters for fixed foveation:
  - Multiview, or single eye per texture: gainX = 4.0f, gainY = 4.0f, foveaArea = 2.0f
  - Two eyes in same texture: double gainX
  - Set minPixelDensity = 0.125f (1:8 quality level)

- Tune for headset and application
  - Again: Always check quality in a headset!



Multiview

CPU

GPU

Generic VR System

Submit Left Draws

Command Buffer

Render Left Eye

1st triangle for left

last triangle for left

left

Submit Right Draws

Command Buffer

Render Right Eye

1st triangle for right

last triangle for right

right

Basic Multiview System

Submit Draw Calls

Command Buffer Left

Render Left Eye

1st triangle for left

last triangle for left

left

Command Buffer Right

Render Right Eye

1st triangle for right

last triangle for right

right

Advanced Multiview System

Submit Draw Calls

Command Buffer

Render Both Eyes

1st triangle for both

amplified

last triangle for both

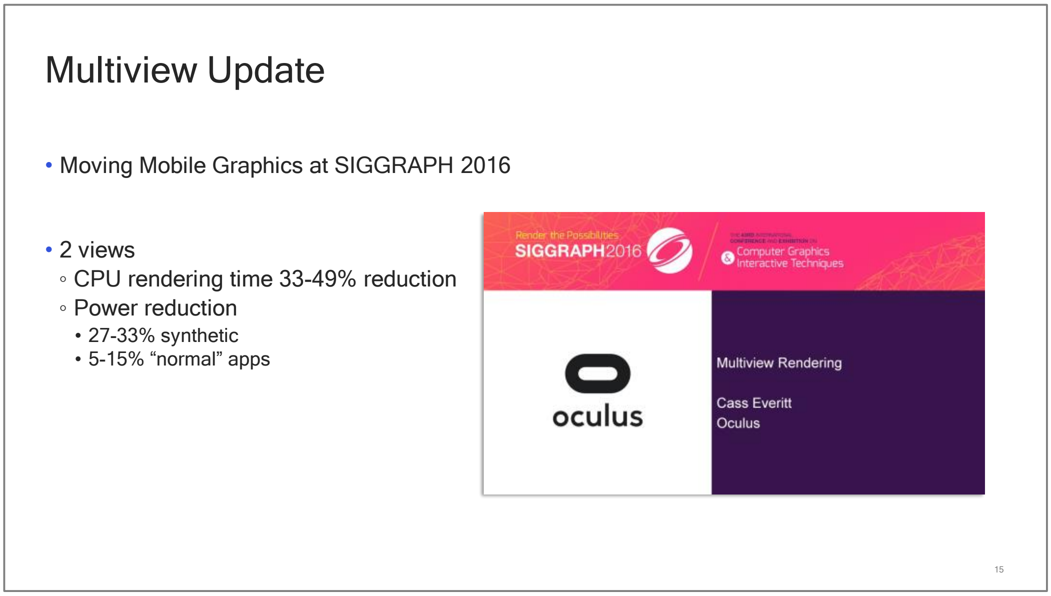
amplified

left

right

14

- How multiview works



Multiview Update

- Moving Mobile Graphics at SIGGRAPH 2016

- 2 views
  - CPU rendering time 33-49% reduction
  - Power reduction
    - 27-33% synthetic
    - 5-15% “normal” apps

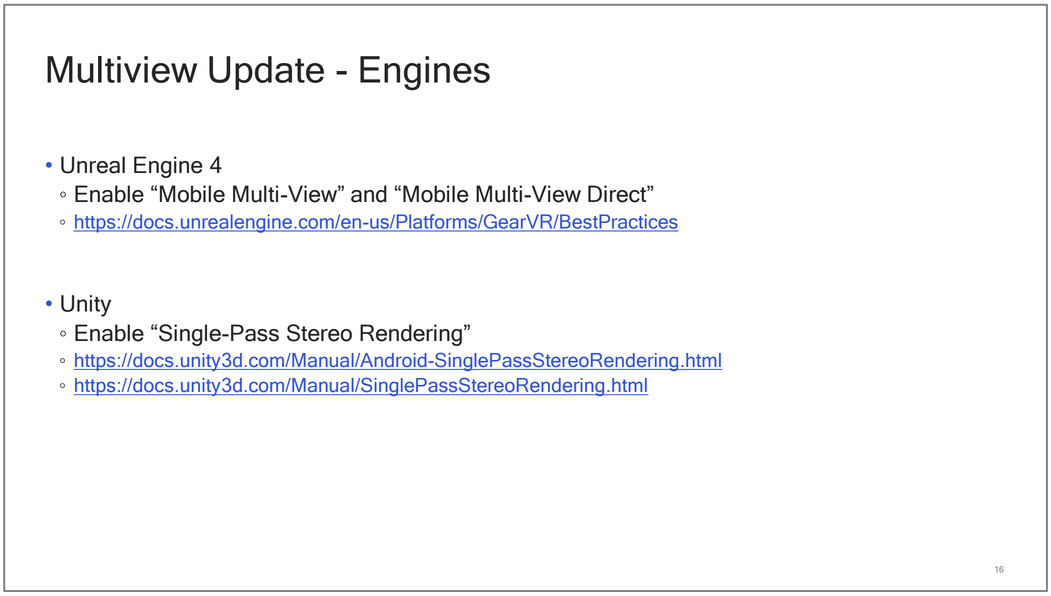
Render the Possibilities
SIGGRAPH2016
THE 43RD INTERNATIONAL CONFERENCE AND EXHIBITION ON
Computer Graphics & Interactive Techniques

oculus

Multiview Rendering

Cass Everitt
Oculus

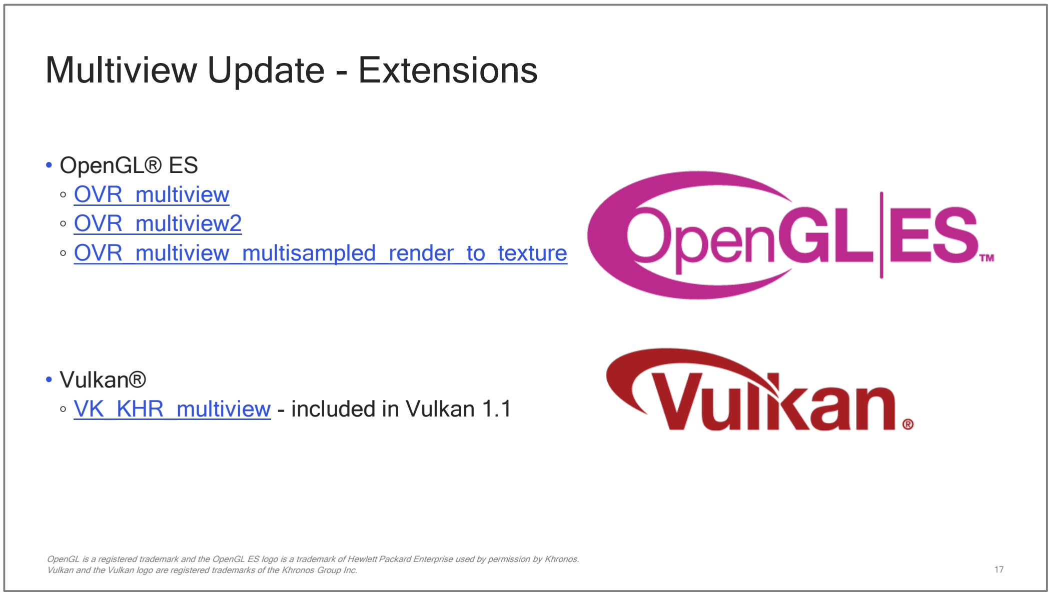
- About the same perf/power today



Multiview Update - Engines

- Unreal Engine 4
  - Enable “Mobile Multi-View” and “Mobile Multi-View Direct”
  - https://docs.unrealengine.com/en-us/Platforms/GearVR/BestPractices
- Unity
  - Enable “Single-Pass Stereo Rendering”
  - https://docs.unity3d.com/Manual/Android-SinglePassStereoRendering.html
  - https://docs.unity3d.com/Manual/SinglePassStereoRendering.html

- Major game engines now have support for multiview
- You may need to do something slightly different depending on VR SDK



Multiview Update - Extensions

- OpenGL® ES
  - OVR_multiview
  - OVR_multiview2
  - OVR_multiview_multisampled_render_to_texture
- Vulkan®
  - VK_KHR_multiview - included in Vulkan 1.1

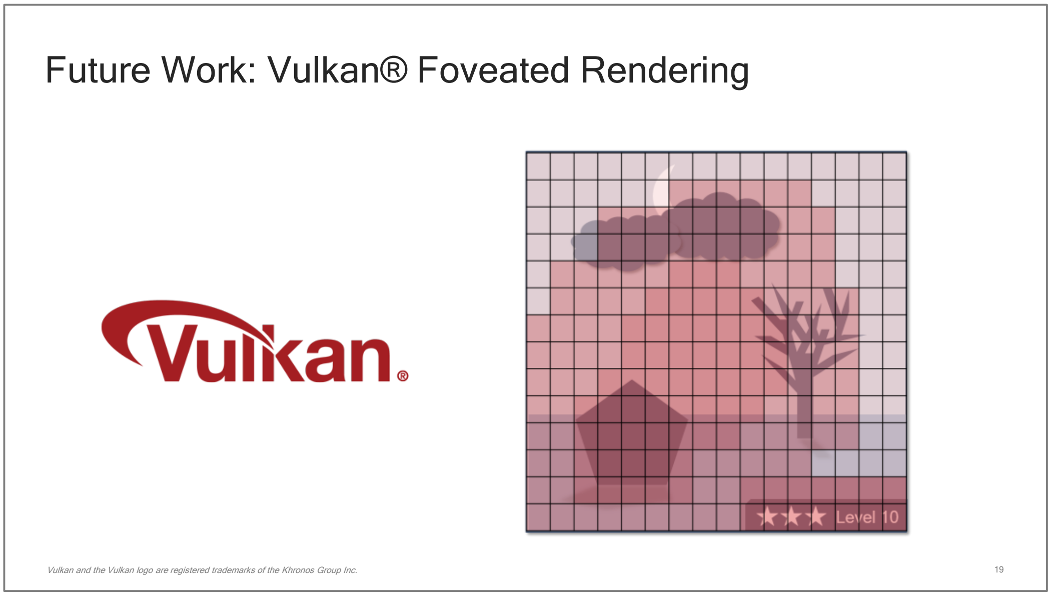
OpenGL is a registered trademark and the OpenGL ES logo is a trademark of Hewlett Packard Enterprise used by permission by Khronos.
Vulkan and the Vulkan logo are registered trademarks of the Khronos Group Inc.

17

- VK_KHR_multiview requires a minimum of 6 views. Interesting use cases for more than 6 views?

Future Work: Eye-Tracking + Foveated Rendering

- Developer Guidelines for Eye-Tracking + Foveated Rendering
  - Qualcomm® Snapdragon™ 845 VR dev kit includes eye-tracking
  - Eye-tracking + foveated rendering enabled in the latest Snapdragon VR SDK

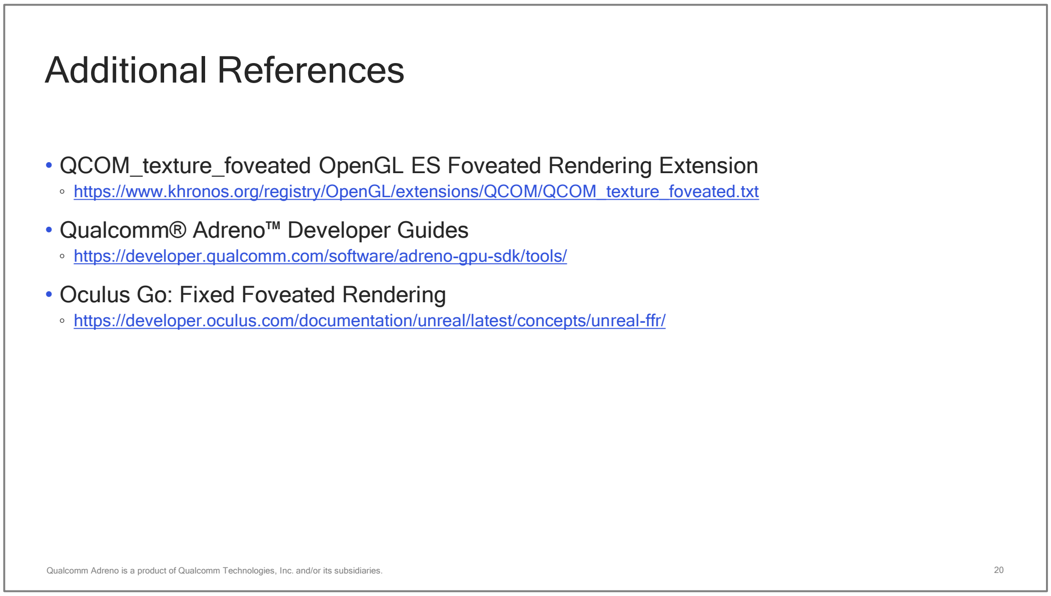


Future Work: Vulkan® Foveated Rendering

Vulkan and the Vulkan logo are registered trademarks of the Khronos Group Inc.

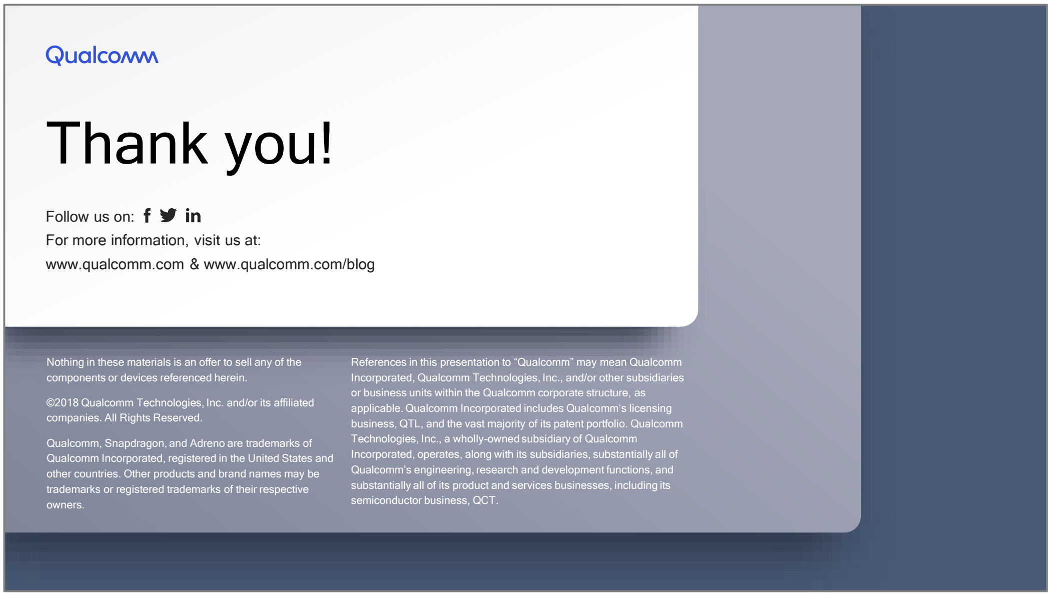
19

- We would like to see a cross-vendor open extension for mobile



Additional References

- QCOM_texture_foveated OpenGL ES Foveated Rendering Extension
  - https://www.khronos.org/registry/OpenGL/extensions/QCOM/QCOM_texture_foveated.txt
- Qualcomm® Adreno™ Developer Guides
  - https://developer.qualcomm.com/software/adreno-gpu-sdk/tools/
- Oculus Go: Fixed Foveated Rendering
  - https://developer.oculus.com/documentation/unreal/latest/concepts/unreal-ffr/



Thank you!

Follow us on: f 🐦 in

For more information, visit us at:
www.qualcomm.com & www.qualcomm.com/blog

Nothing in these materials is an offer to sell any of the components or devices referenced herein.

References in this presentation to "Qualcomm" may mean Qualcomm Incorporated, Qualcomm Technologies, Inc., and/or other subsidiaries or business units within the Qualcomm corporate structure, as applicable. Qualcomm Incorporated includes Qualcomm's licensing business, QTL, and the vast majority of its patent portfolio. Qualcomm Technologies, Inc., a wholly-owned subsidiary of Qualcomm Incorporated, operates, along with its subsidiaries, substantially all of Qualcomm's engineering, research and development functions, and substantially all of its product and services businesses, including its semiconductor business, QCT.