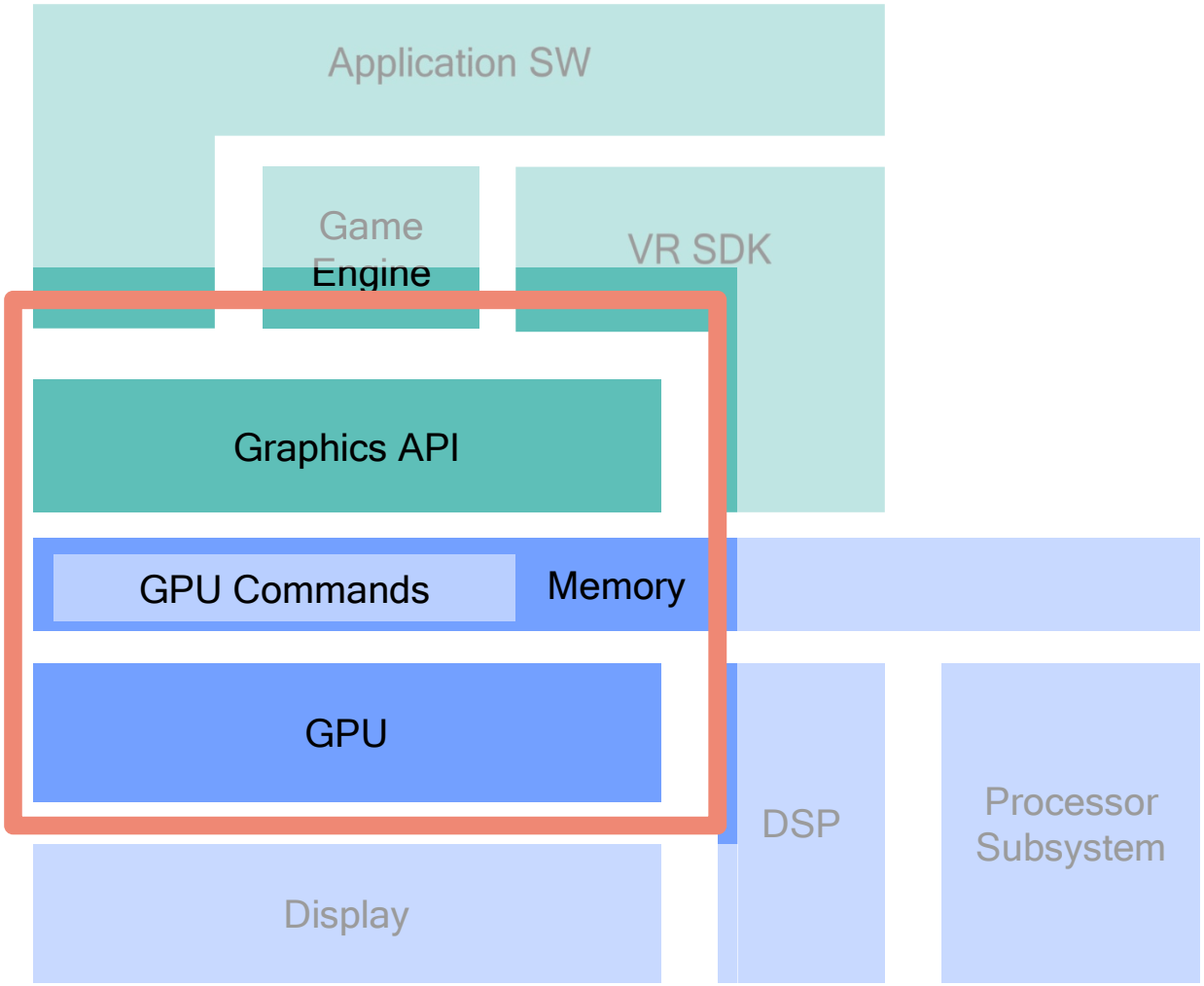
Mobile Considerations for XR

Reality, Virtually, Hackathon!

Chris Kolb, Senior Director, Engineering
Qualcomm Technologies, Inc.
October 2017



Introductions

Chris Kolb

Director of SW Eng
GPU driver SW
Qualcomm Technologies, Inc.

Sam Holmes

Staff SW Eng
GPU driver Dev
Specializing in XR
Qualcomm Technologies, Inc.

VR Vocabulary

Eye Buffers – These are buffers rendered by game engines for the left and right eyes

Asynchronous Reprojection – This is a separate asynchronous rendering thread run on the GPU to correct lens distortion and reduce latency.

DoF – Degrees of Freedom. In 3 DoF, position tracking only accounts for rotations. In 6 DoF, position tracking accounts for both rotations and translations

VR Pipeline – GPU workload

Asynchronous reprojection requires low latency/ quick response time

Stereo Rendering

Async. Reprojection

Display

Brute force rendering requires very high GPU horsepower for desired FPS

Large resolutions result in huge DDR traffic

Command Buffer

Rendering L&R Views

Binning Rendering (Bin 0) ... Rendering (Bin N)

Barrel Distorted mesh

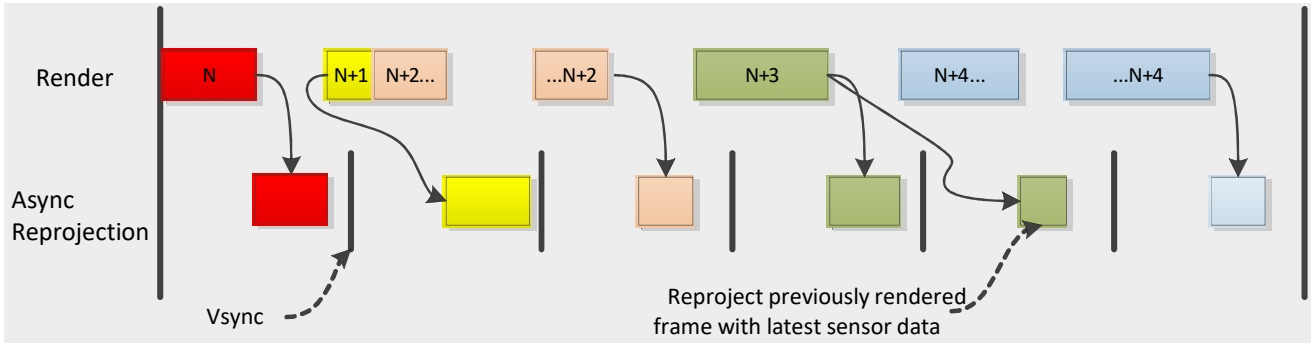
Distortion

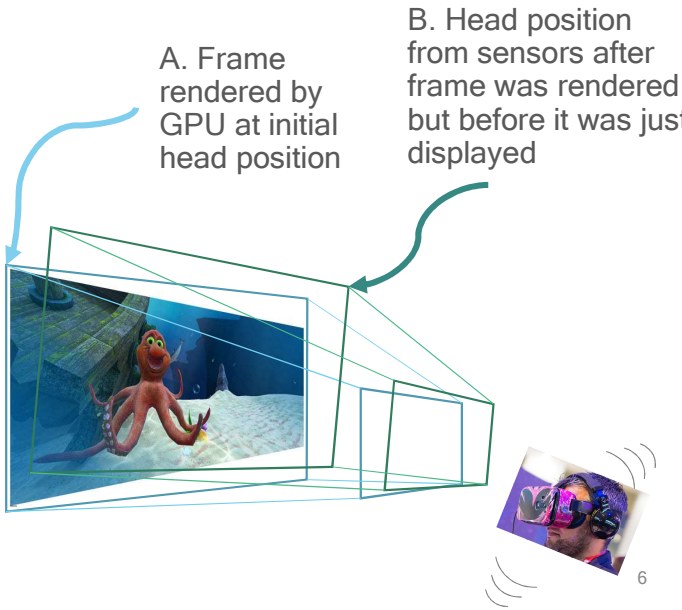
Asynchronous Reprojection, Distortion, Color Correction, Composition

Asynchronous Reprojection

Asynchronous Reprojection Explained

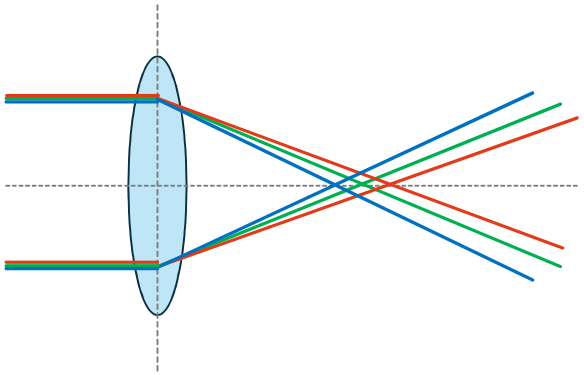
- Asynchronous Reprojection is a *technique* applied in VR to optimize the Motion to display timing
  - Eye buffer rendered at a slightly larger than required resolution
  - Reprojected based on latest head pose just before display. (predictable in time)
- Reducing latency: Motion-to-photon
  - At 60fps, rendered frame can be 30-50ms stale w.r.t head/eye position when displayed
  - With reprojection Motion-to-photon < 20ms
- Can compensates for missed frames by reprojecting previous rendered frame to correct for updated head position



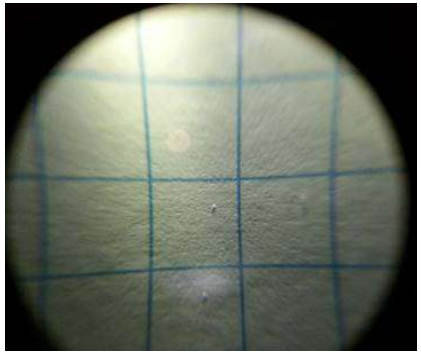


Chromatic Aberration Correction and Barrel Distortion

- Color Aberration Correction (CAC) is applied to the Red, Green and Blue channels independently to correct for the lens effects at different wavelength of light
- Significant BW requirements
  - Requires three fetches (one for R, one for G, and one for B)

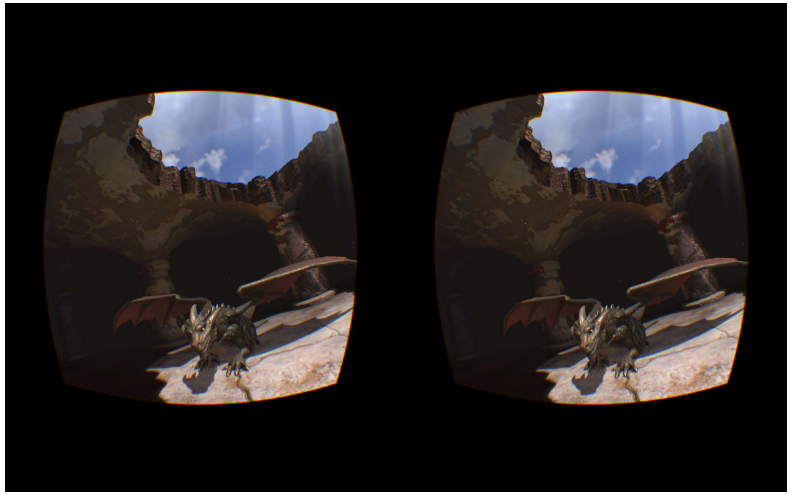


Chromatic Aberration



Pin Cushion Distortion

- Barrel Distortion is applied to the image to counter the VR wide angle lens pin cushion distortion.



Barrel Distortion

Foveated Rendering

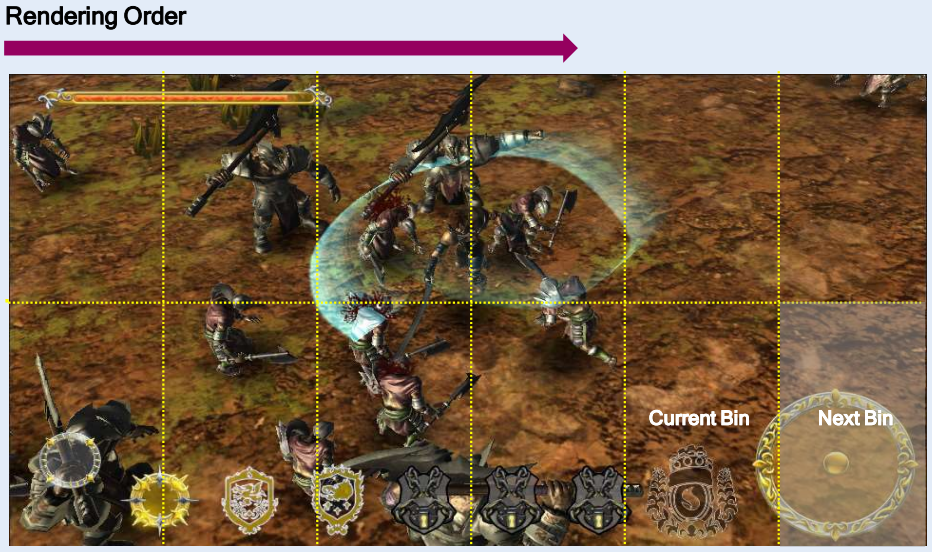
Tile Based Rendering

Rendering directly to the frame buffer in system memory is costly in Mobile

- Limited bandwidth to the frame buffer
- DDR power consumption

Solution = Tile Based Rendering

- Rather than render the scene in a single pass, render in multiple passes
- Each pass renders a complete tile to on-chip memory



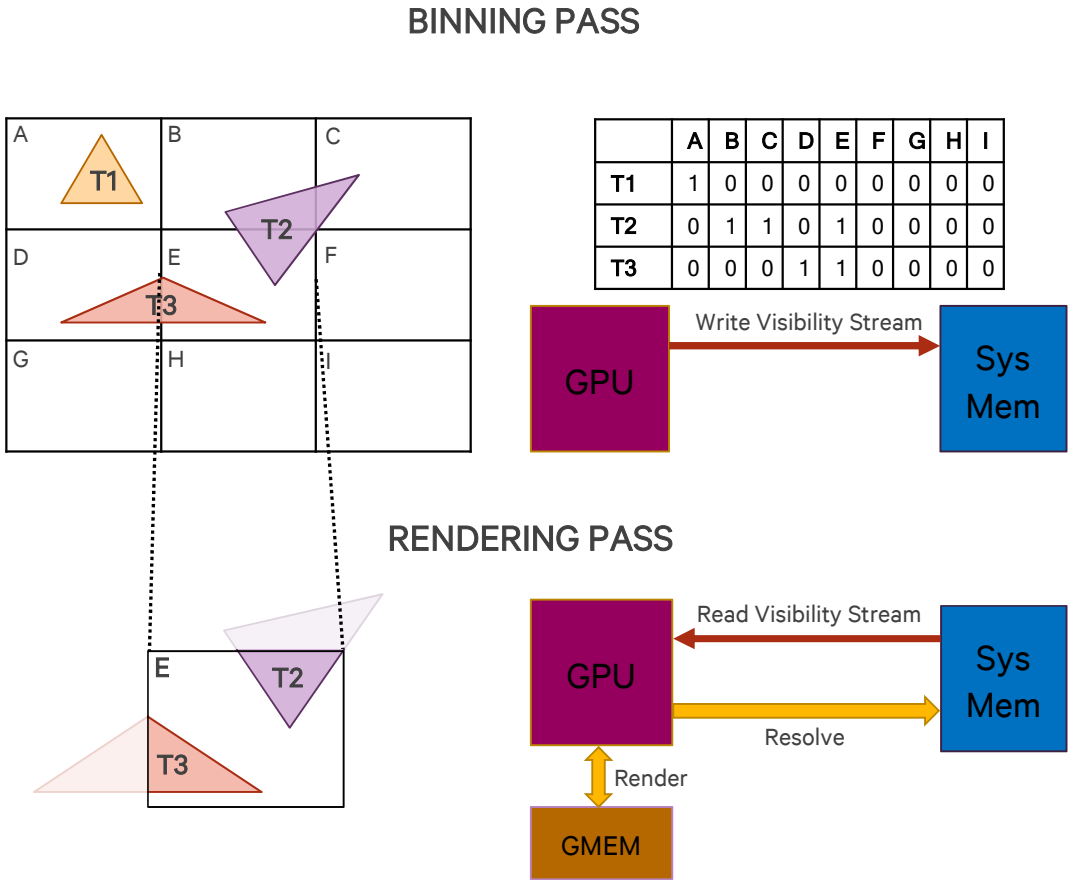
Overview of Binning/Tiling Architecture

Binning requires multiple passes over the graphics primitives

Binning Pass
Mark bins where a triangle is visible (visibility stream)

Rendering Pass
Processes primitives per bin by reading the visibility stream

When rendering is complete, the results are copied from graphics memory (GMEM) on the GPU to the system memory



	A	B	C	D	E	F	G	H	I
T1	1	0	0	0	0	0	0	0	0
T2	0	1	1	0	1	0	0	0	0
T3	0	0	0	1	1	0	0	0	0

Foveated Rendering

Fovea

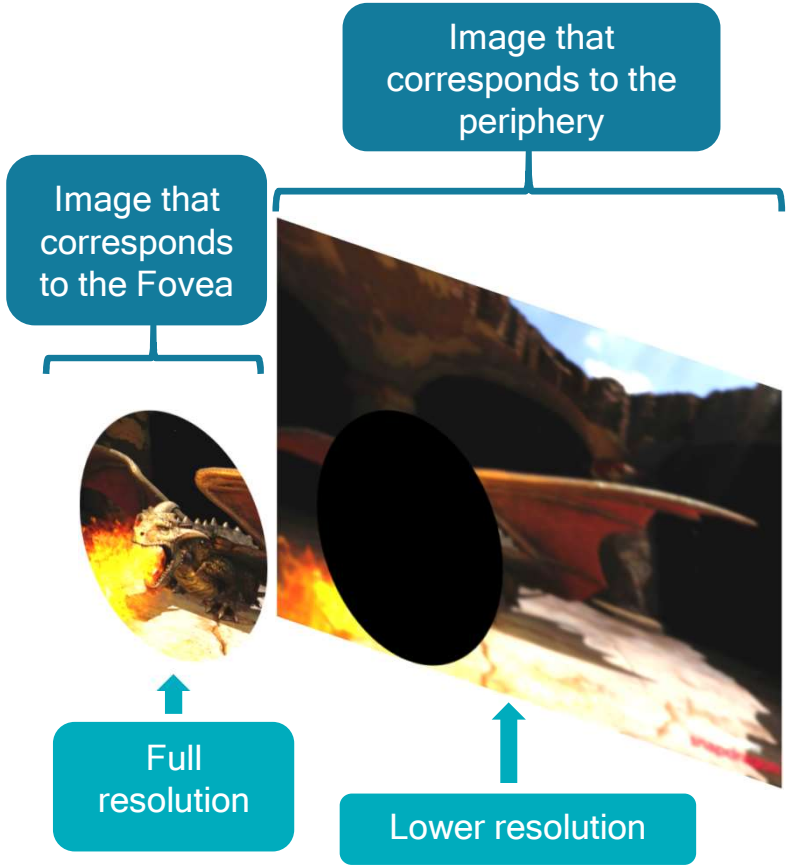
- The human eye only sees highest resolution at the center

Foveated Rendering

- Only render at higher resolution where the eye is pointing
- Render at lower resolution elsewhere

Advantages

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



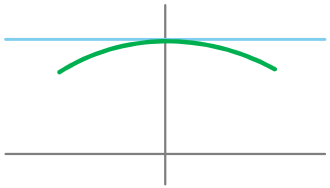
Fixed (Lens based) vs Eye Tracking based Foveation

In absence of eye tracking, fixed foveation is still possible due to lens distortion at the edges

Fixed (Lens Based) Foveation

- Lens distortion causes pixels on the periphery to be compressed
- Lenses have poor quality
- Hence, render highest resolution at lens center and low resolution at the periphery
- Center of foveation is fixed to lens center
- Quality fall-off from center is moderate and depends on lens characteristics and content

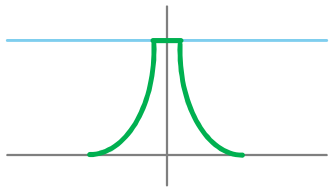
Lens Distortion
Moderate quality falloff



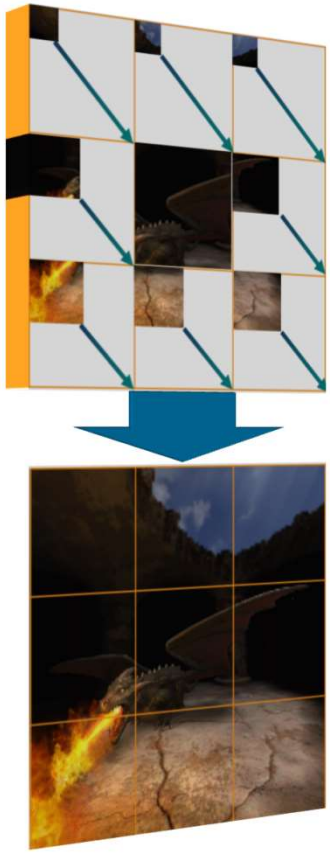
Eye Tracking Based Foveation

- Human eye only sees highest resolution at the center
- Render at highest resolution only where the eye is pointing and lower resolution elsewhere
- Center of foveation tracks with eye movements
- Quality fall-off is steep and depends on eye characteristics

Eye Tracking
Steep quality falloff



Bin Based Foveated Rendering



The framebuffer is broken into tiles that are rendered individually.

While rendering each tile, the viewport area is downscaled based on application provided parameters.

When each tile is resolved from graphics memory it is upscaled to its original area on the full resolution surface.

An application can now use the full resolution surfaces for post processing or displaying on screen

Mobile Performance

Mobile users want performance

SOURCE: *FAILING TO MEET MOBILE APP USER EXPECTATIONS: A MOBILE APP USER SURVEY*, DIMENSIONAL RESEARCH, FEBRUARY 2015

96% of mobile users consider performance important

SOURCE: *FAILING TO MEET MOBILE APP USER EXPECTATIONS: A MOBILE APP USER SURVEY*, DIMENSIONAL RESEARCH, FEBRUARY 2015

58% of mobile users expect an app to load in 2 seconds or less

SOURCE: *NEUMOB SURVEY*, DECEMBER 2016

75% of mobile users will delete an app because of poor performance

SOURCE: *NEUMOB SURVEY*, DECEMBER 2016

Performance = Smooth interface + Usability

SOURCE: *NEUMOB SURVEY*, DECEMBER 2016

Performance = Smooth interface + Usability

+ Power Consumption

+ Device Temperature

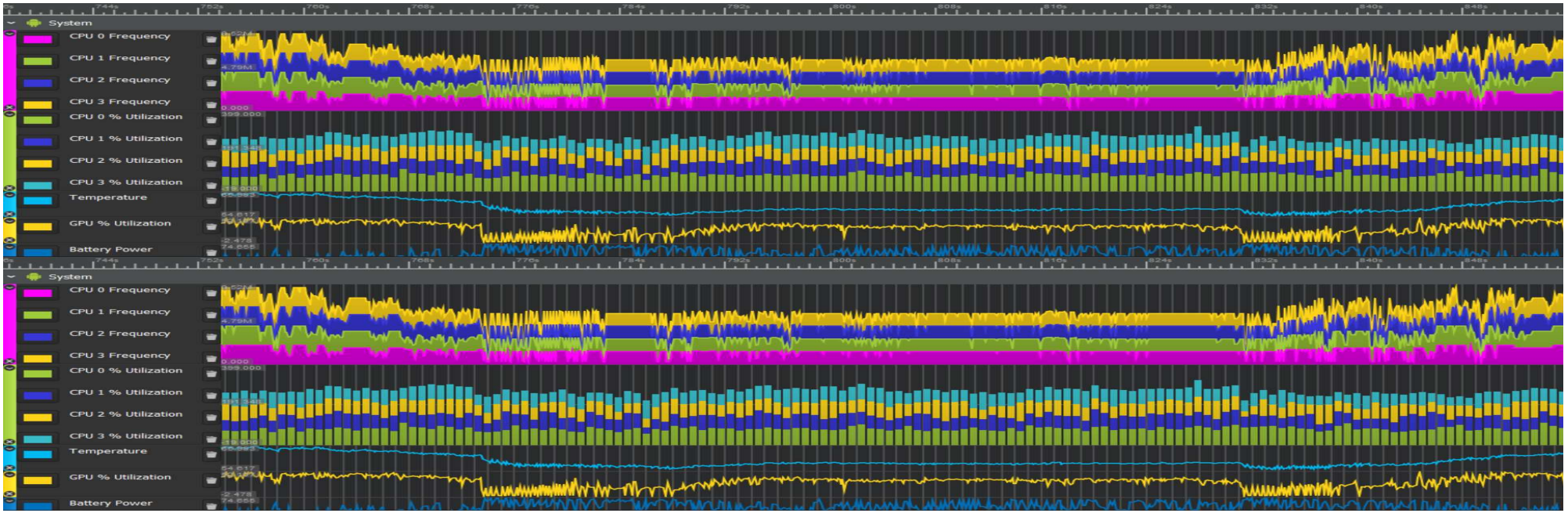
SOURCE: *NEUMOB SURVEY*, DECEMBER 2016

“Even if you consistently hold 60 FPS, more aggressive drawing consumes more battery power, and subtle improvements in visual quality generally aren’t worth taking 20 minutes off the battery life for a title.

- Oculus Mobile SDK Documentation

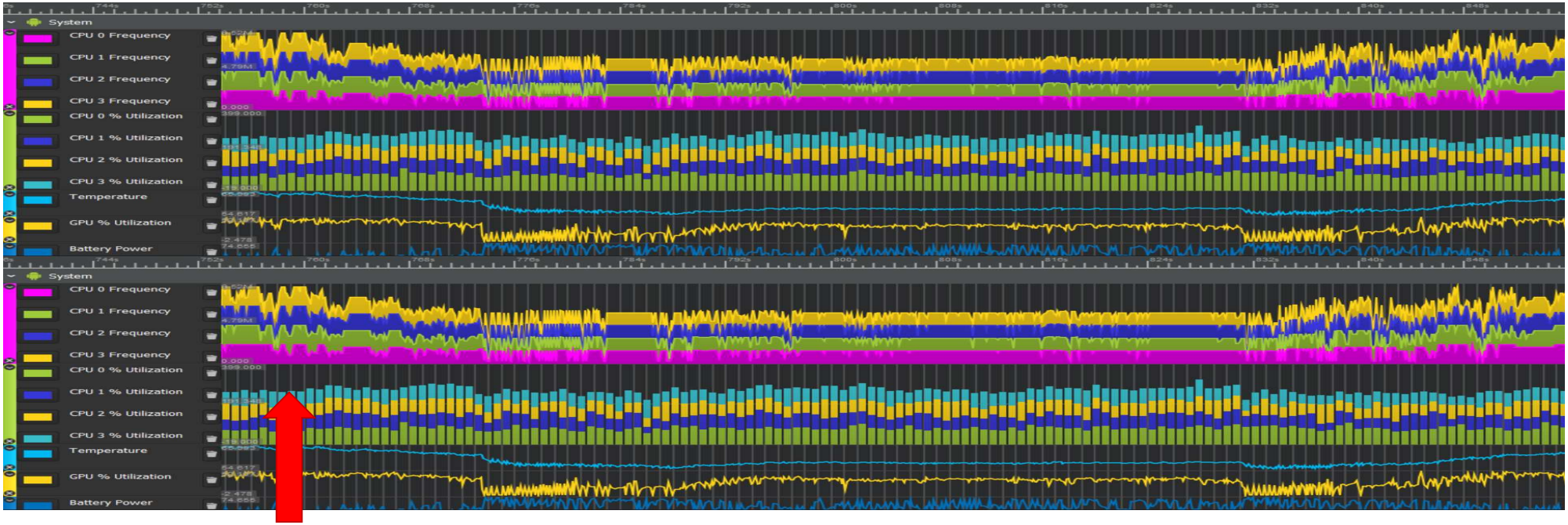
System Performance

What happens when Temperature gets too high?



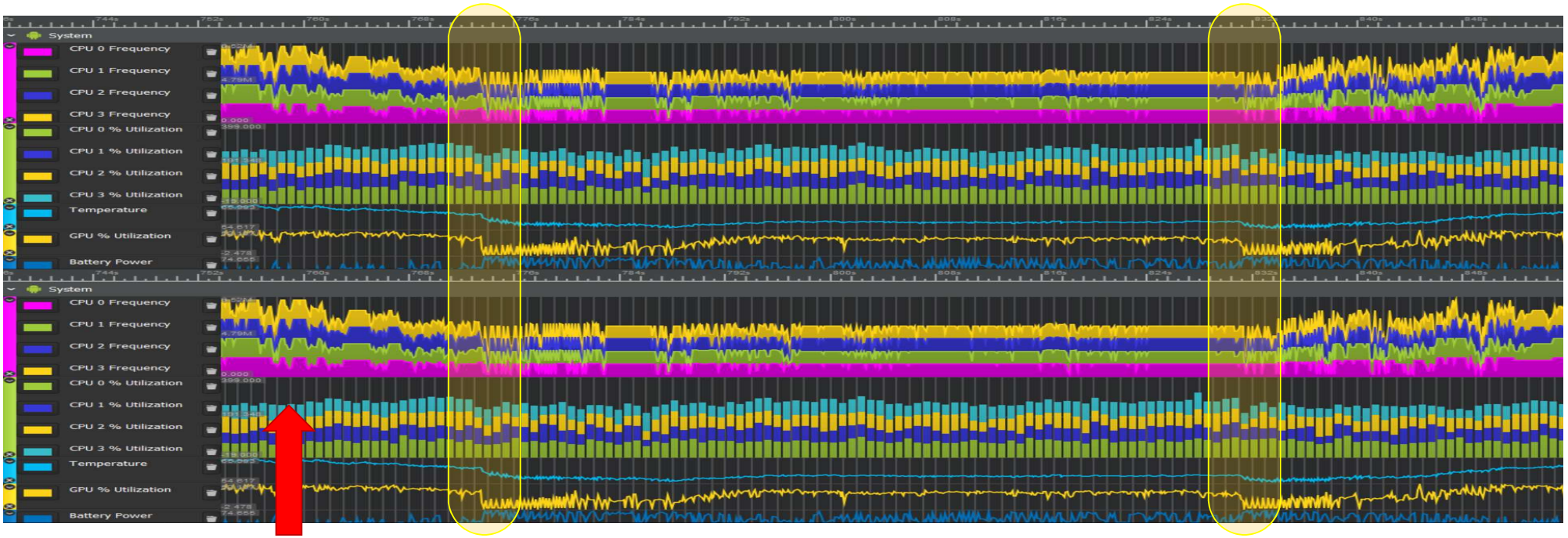
System Performance

What happens when Temperature gets too high?



System Performance

What happens when Temperature gets too high?



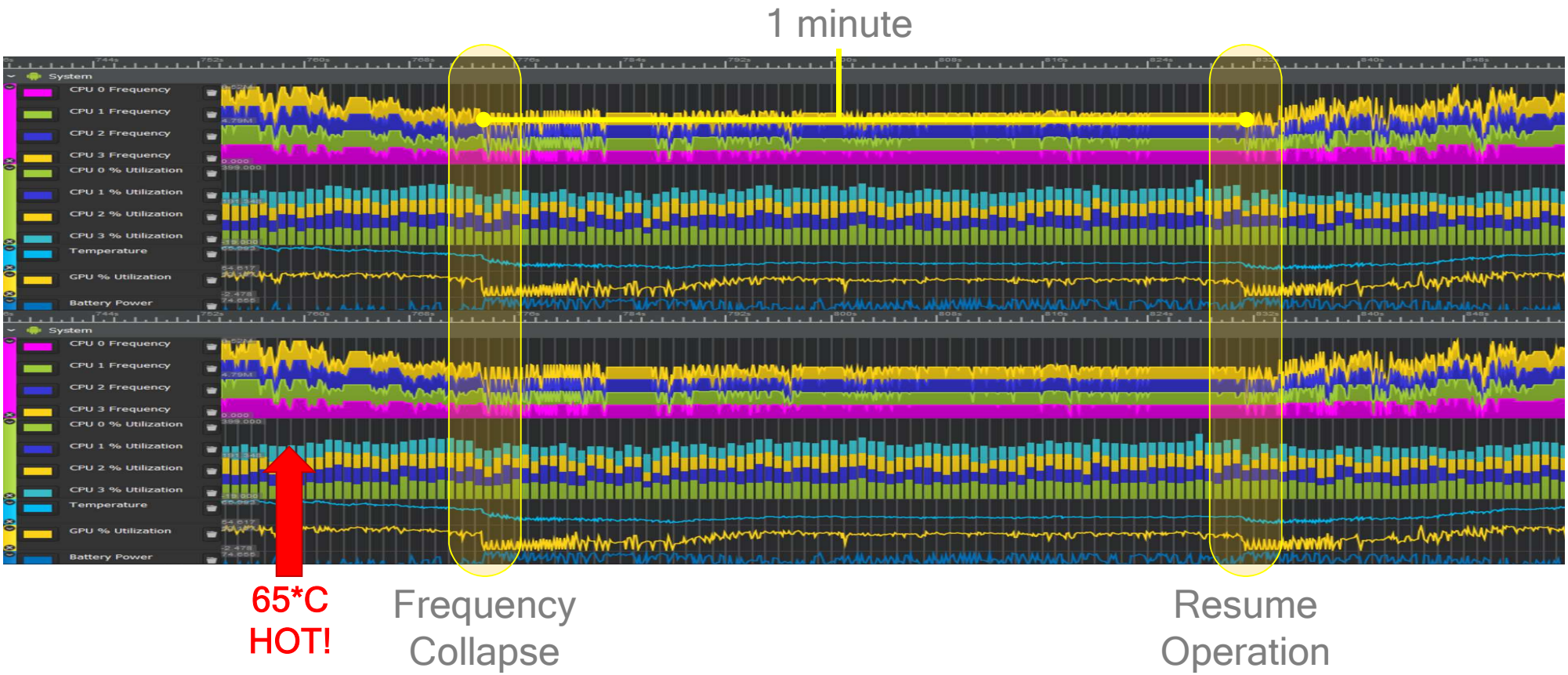
65*C
HOT!

Frequency
Collapse

Resume
Operation

System Performance

What happens when Temperature gets too high?



VR Optimizations & Best Practices

Keep Content Complexity in Check

Mobile

- 60 FPS (required by Oculus)
- 50-100 draw calls per frame
- 50,000-100,000 triangles or vertices per frame

general guidelines for establishing your baselines, given as approximate ranges

- Oculus Mobile SDK Documentation

Developed Mobile Games before?
What changes for VR

In VR, missing frames leads to motion sickness

- Much worse than just some stuttering or janks

VR needs steady 60 fps display refresh

VR application development different than smartphone games

- More platform specific code
- Many more system level considerations with much tighter requirements
- Applications should be “Async Reprojection friendly”
- Must develop on HMDs

VR has Async Reprojection running in background

- 3 extra full screen Memory transfers – power and system bandwidth implications
- High priority GPU jobs that must run at display refresh rate
- CPU must have time to schedule this asynchronous workload

More system level functions running (sensor hub) which have power implications

Key Goals for VR on Mobile

Thermally sustainable workload

- Applications need to run for 20 mins to 2 hours with NO throttling
- Throttling will lead to janks and late reprojection – this causes motion sickness

Motion to photon latency

- The amount of time taken from head/body motion to screen update
- Anything over 20ms causes motion sickness

Usable CPU/GPU/Bandwidth frequencies

Max screen resolution and framerate

Max eyebuffer resolution and framerate

VR Best Practices

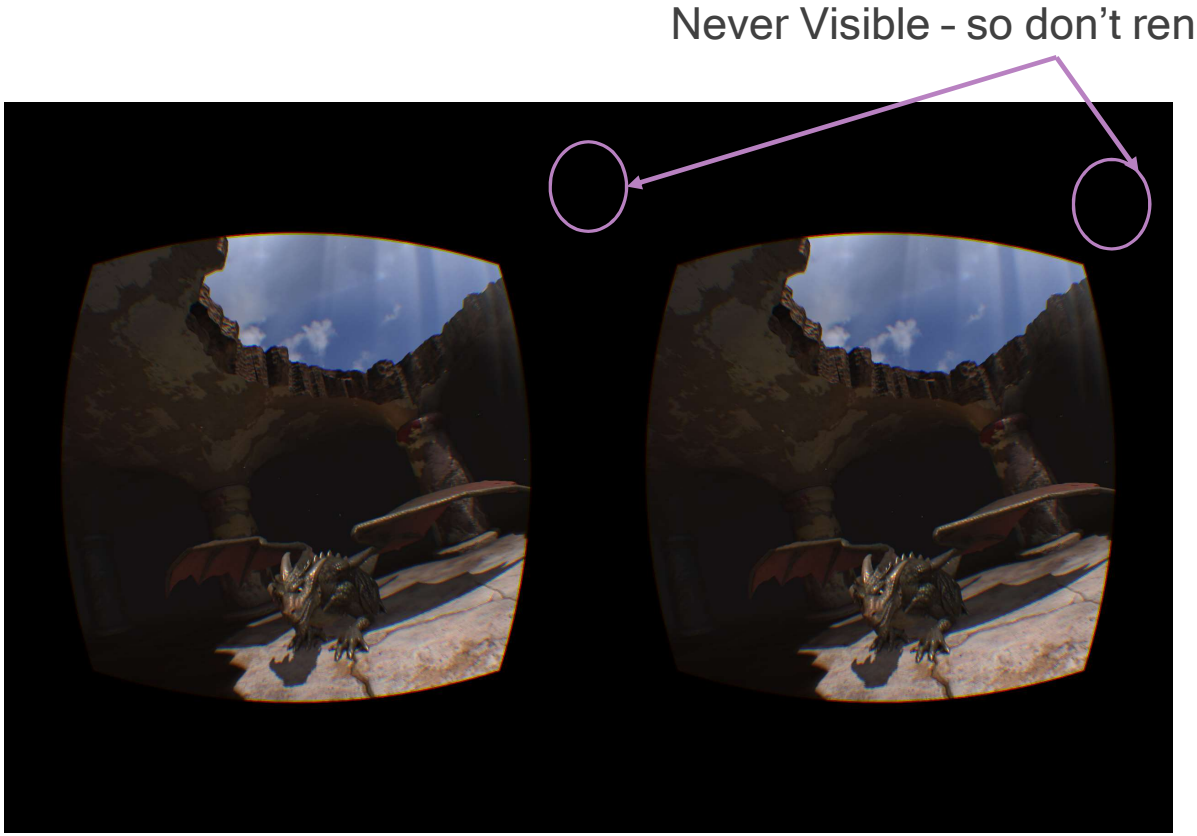
- User comfort is paramount
  - Applications should render at display frequency (60 fps)
    - Remember to budget ~2ms for Async Reprojection
    - Do not assume applications can run at 30Hz on a 60Hz display and have async reprojection make up the difference
    - Async Reprojection is a great “insurance” policy for an occasionally dropped frame, and further reduces latency
  - When simulating user movement through a virtual space, do so at a constant velocity.
    - Varying or abrupt changes to velocity can be nauseating in VR
- Traditional toolboxes of non-stereo rendering techniques may no longer work as expected in VR
  - Only the viewpoint should differ between eyes. Rendering techniques which do not generate the same results between eyes should be avoided
  - Normal maps will tend to flatten out and the illusion of depth they create will fail when viewed in stereo.
    - Consider utilizing additional geometry or parallax mapping to add detail to objects that will be very close to the user in VR
  - Large camera aligned billboards (often generated by particle systems) do not work well when viewed in stereo VR and should be avoided
  - Skybox planes can become apparent in stereo VR
    - Scale the skybox to maximum scale to reduce the effect

VR Best Practices

- MSAA is extremely important for VR
  - 2x MSAA is the minimum that should be used, 4x MSAA is preferred
- Power and thermal are especially important for VR
  - Dropped frames and judder introduced by thermal mitigation can lead to physical sickness in users
  - Design applications with thermal and balanced performance in mind
  - Extensive use of tools like Qualcomm® Snapdragon™ Profiler throughout the development cycle are essential for measuring and improving application performance

Optimizations

Some Portions of the Screen are Never Seen



Using beginEye / endEye writes out the stencil buffer so that the rendering is optimized in the eye buffer and reprojection stages based on the stencil defined in the device config.

VR Application – Buffer Writes

"Avoid bin fills from main memory and unnecessary buffer writes"

- Oculus Developer Guide, Mobile SDK

GMEM Loads and Stores

- Also referred to as Unresolves / Resolves
- Moves memory from system memory to tile memory and back
- To avoid loads. Whenever possible
  - Clear or glInvalidateFramebuffer/glDiscardFramebufferEXT
  - Ensure all attachments are cleared, including depth + stencil
- Discard or Invalidate any surfaces you don't need to Store
  - App can call discard/invalidate prior to flush
  - Discard depth/stencil if not used by async reprojection
  - Especially true if using the EXT_multisampled_render_to_texture extension which leaves depth buffers undefined
  - Qualcomm® Snapdragon™ VR SDK: Use beginEye()/endEye()

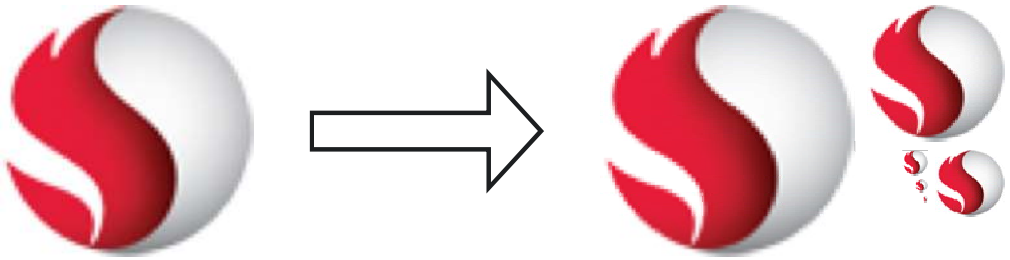
Case Study: VR Application - Mipmaps

"glGenerateMipMaps() is fast and efficient; you should build mipmaps even for dynamic textures (and of course for static textures)"

- Oculus Developer Guide, Mobile SDK

Case Study: VR Application - Mipmaps

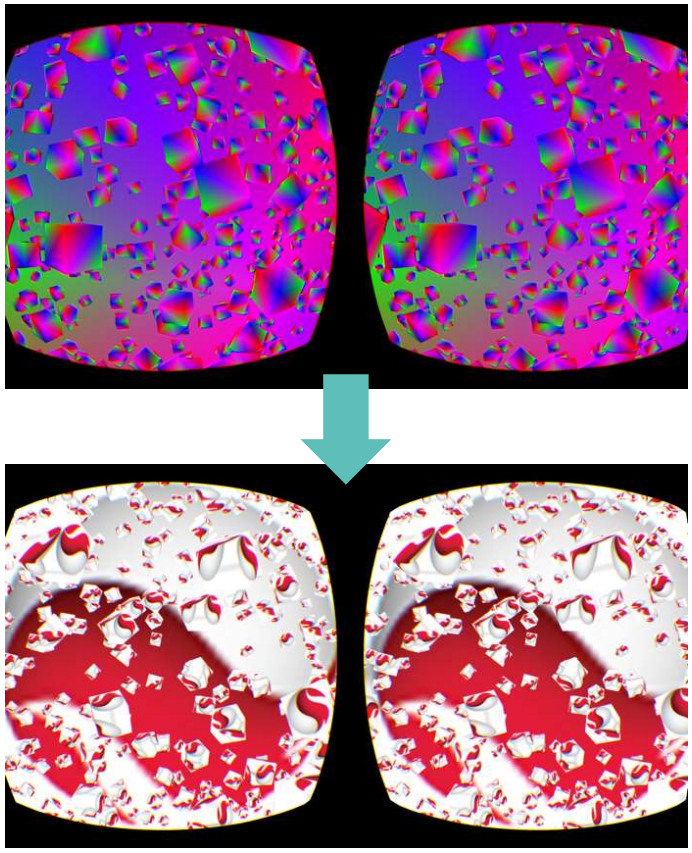
Definition



Case Study: VR Application – Mipmaps

Setup

- Modified sample app from Snapdragon™ VR SDK
- Device: Samsung Galaxy S7 (Snapdragon 820 processor)
- Added 1k x 1k uncompressed texture to all cubes
- 2 versions
  - Raw texture
  - Generated mipmaps



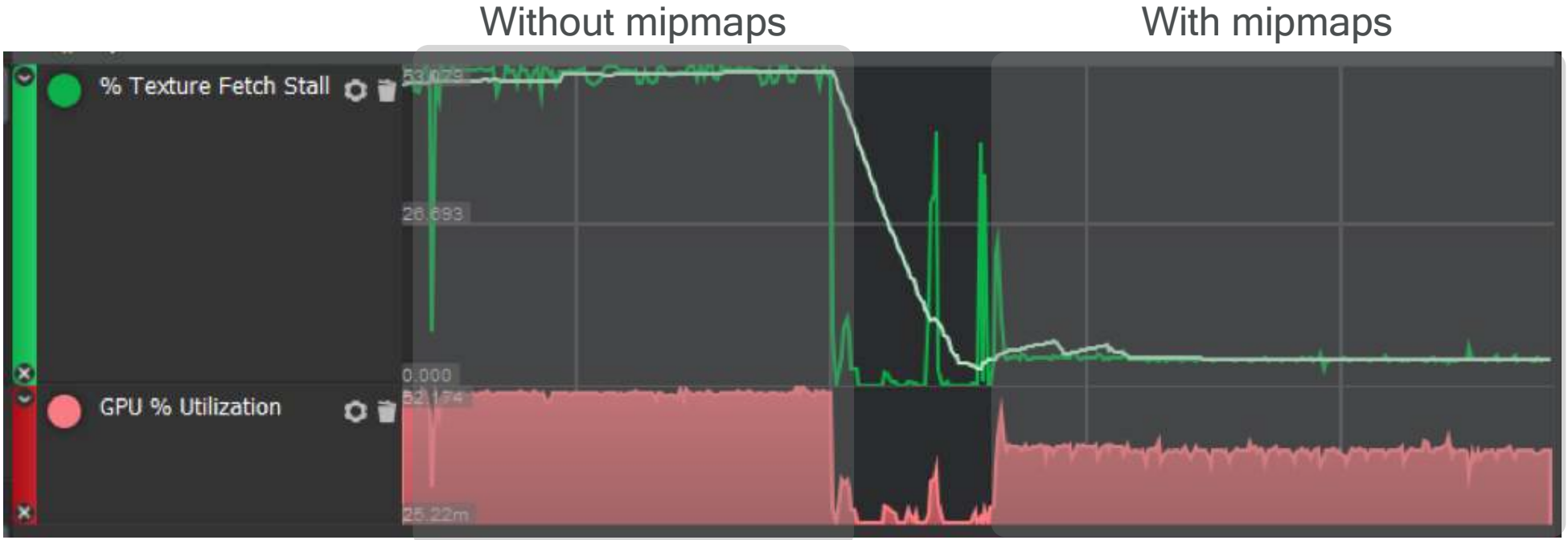
Case Study: VR Application – Mipmaps

Analyzing App without Mipmaps

GPU is spending half its time stalled while fetching texture data

Case Study: VR Application – Mipmaps

Analyzing with and without Mipmaps



Using Mipmaps – reduced the GPU texture stalls by 80%

- Reduced per eye render time by 48%

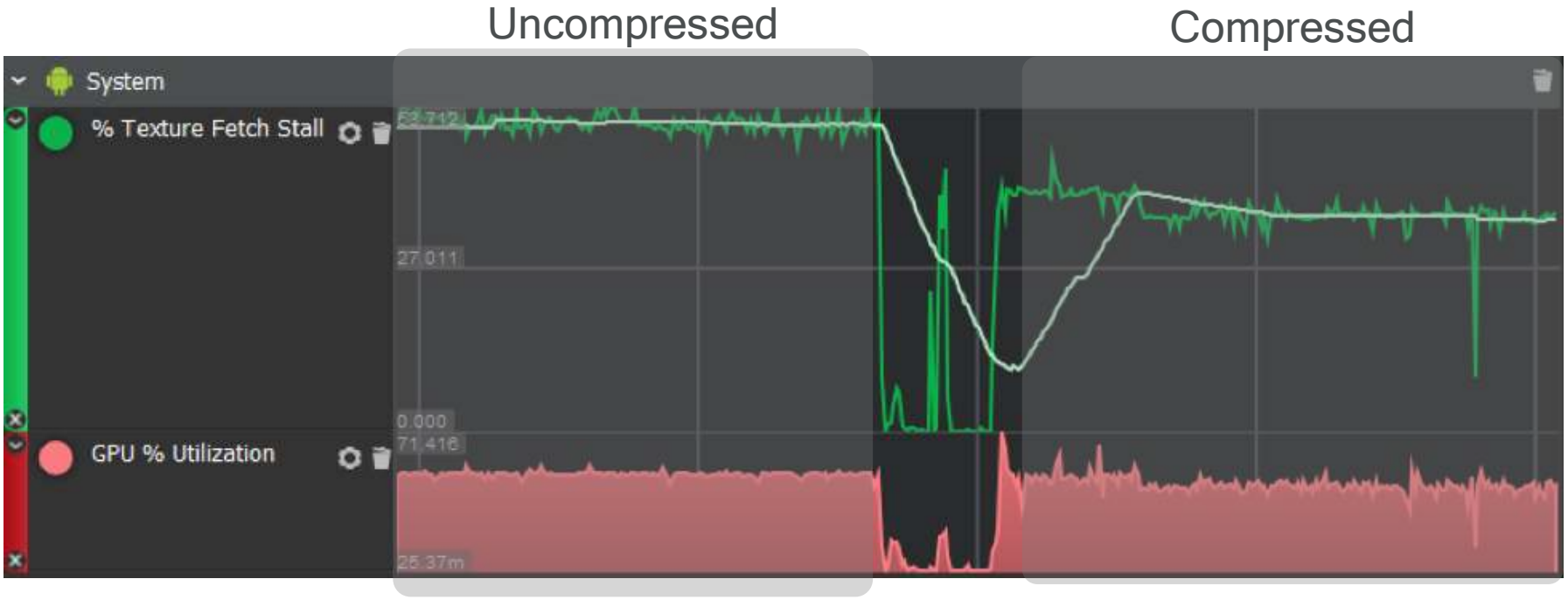
VR Application – Texture Compression

"Texture compression offers significant performance benefits"

- Oculus Developer Guide, Mobile SDK

VR Application – Texture Compression

Analyzing App – Comparison



VR Optimization Checklist

- Application or Engine or Both
  - Discard unused buffers (eg. Depth) and clear/invalidate to avoid loads (Both)
  - Use reasonable CPU/GPU workloads to make a thermally stable application (App)
  - Don’t render to portions of eye buffer that aren’t visible (Engine/SDK)
  - Use Foveation to reduce pixel load and bandwidth (Engine)
  - Use Multiview to reduce stereo rendering workloads (Engine)
  - Use MSAA (App/Engine)
  - Use forward rendering algorithms (not deferred shading) (Engine setting)
  - Use short shaders (App/Engine)
  - Keep post processing effects to a minimum (App)
  - Reserve ~20% GPU performance (2+ ms) for the async reprojection operation (App)

Resources

Resources

Qualcomm Developer Network – https://developer.qualcomm.com

Snapdragon Profiler – Available from Qualcomm Developer Network
https://developer.qualcomm.com/software/snapdragon-profiler

- Versions available for Windows, macOS, and Linux
- Tutorial Videos
- FAQs
- Release Notes
- Forum

Snapdragon VR SDK (download via Thundercomm)
https://developer.qualcomm.com/download/snapdragon-sdk/snapdragon-vr-sdk

Adreno OpenGL ES Developer Guide - Available from Qualcomm Developer Network
https://developer.qualcomm.com/download/adrenosdk/adreno-opengl-es-developer-guide.pdf

Thank you

Follow us on:

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