GDC

Advanced VR Rendering Performance

Alex Vlachos, Valve
Alex@ValveSoftware.com

UBM

My Presentation Last Year

- This is part 2 of my presentation from last year “Advanced VR Rendering”, GDC 2015
- Video and slides from last year are free online: http://www.gdcvault.com/play/1021771/Advanced-VR
- This year’s talk focuses on performance, but increased visual quality is the goal of what I’m presenting today

Outline

VALVE

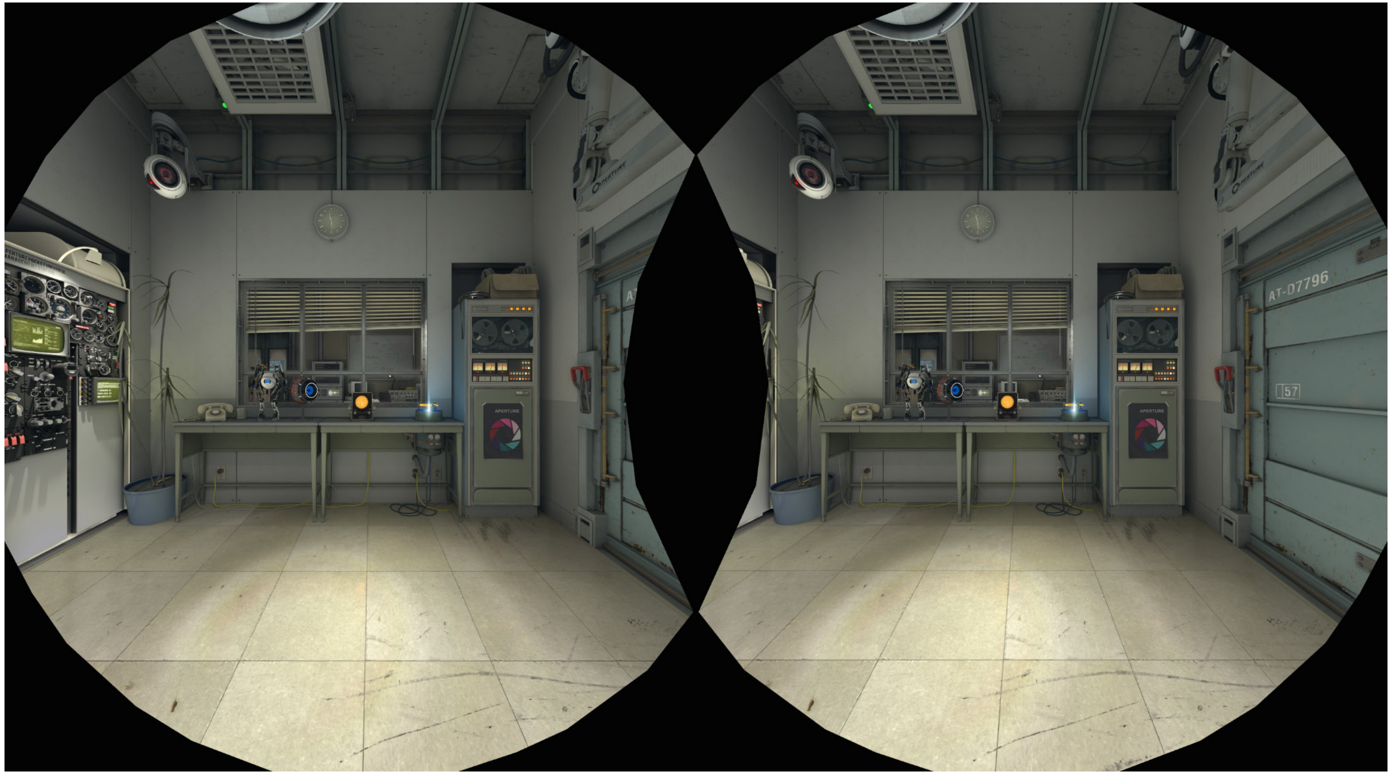
- Multi-GPU for VR
- Fixed Foveated Rendering & Radial Density Masking
- Reprojection
- Adaptive Quality

Recap: Hidden Area Mesh

AT-D7796
57
APERTURE

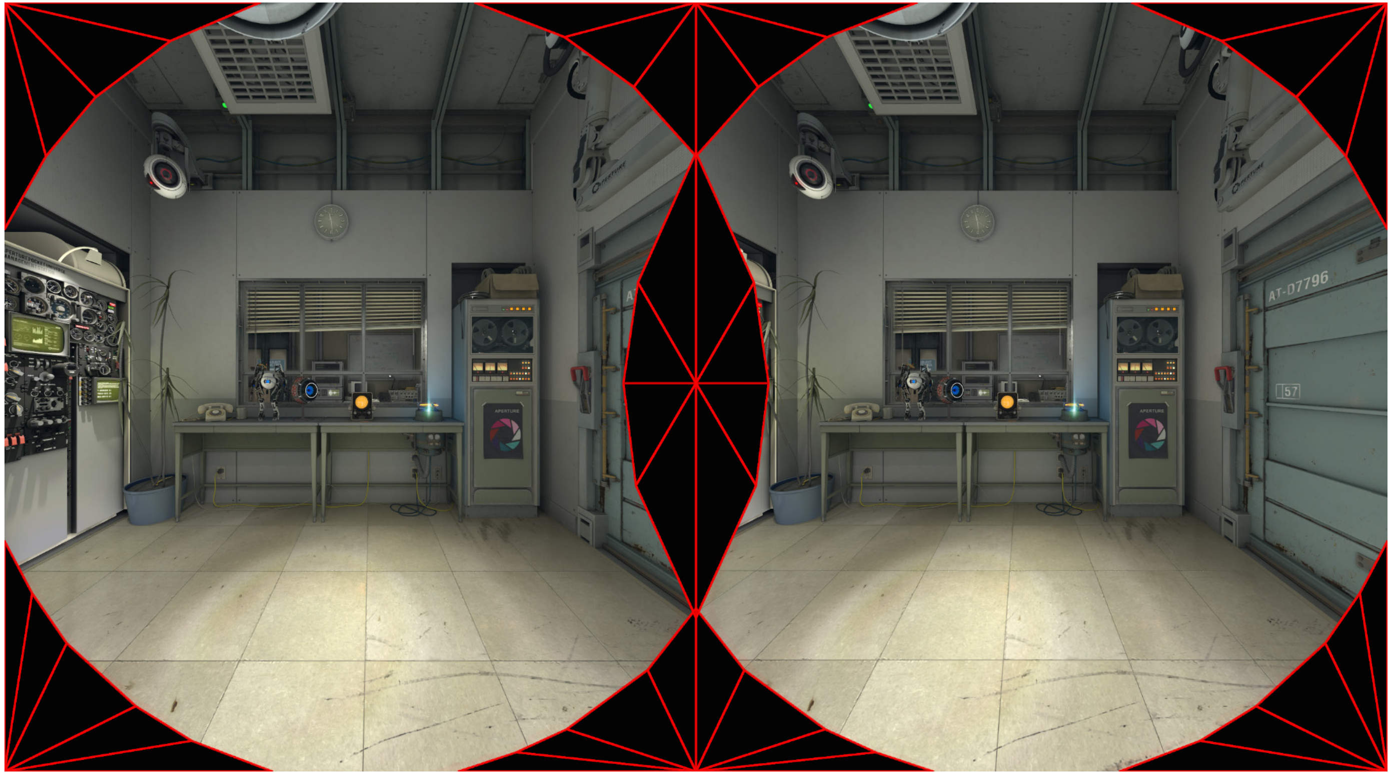
(From “Advanced VR Rendering” GDC 2015)

Recap: Hidden Area Mesh



(From "Advanced VR Rendering" GDC 2015)

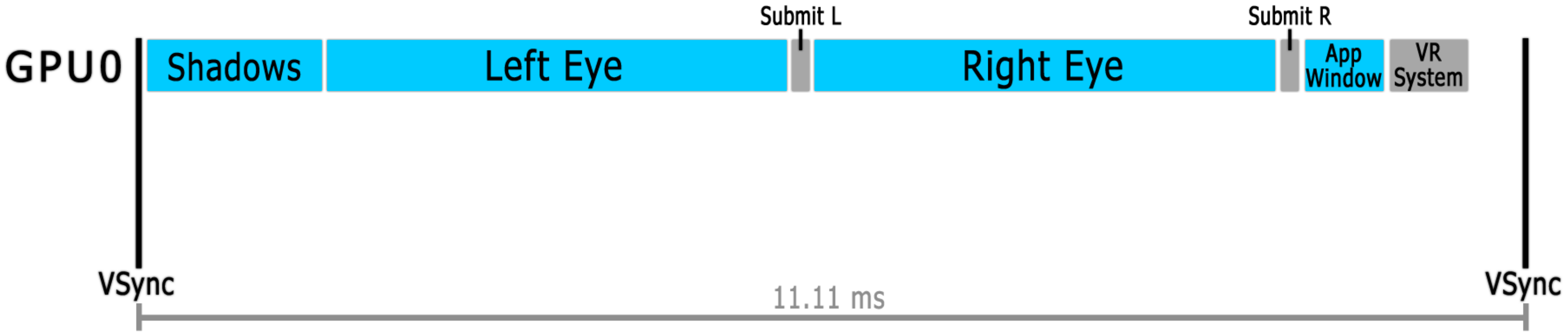
Recap: Hidden Area Mesh



(From "Advanced VR Rendering" GDC 2015)

Single GPU

- Single GPU does all the work
- Stereo rendering can be accomplished a number of ways (this example uses sequential rendering)
- Shadow buffer is shared by both eyes

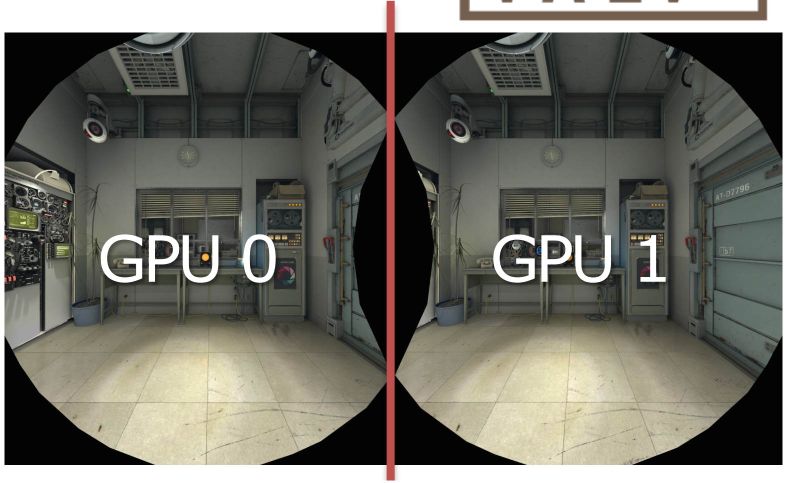


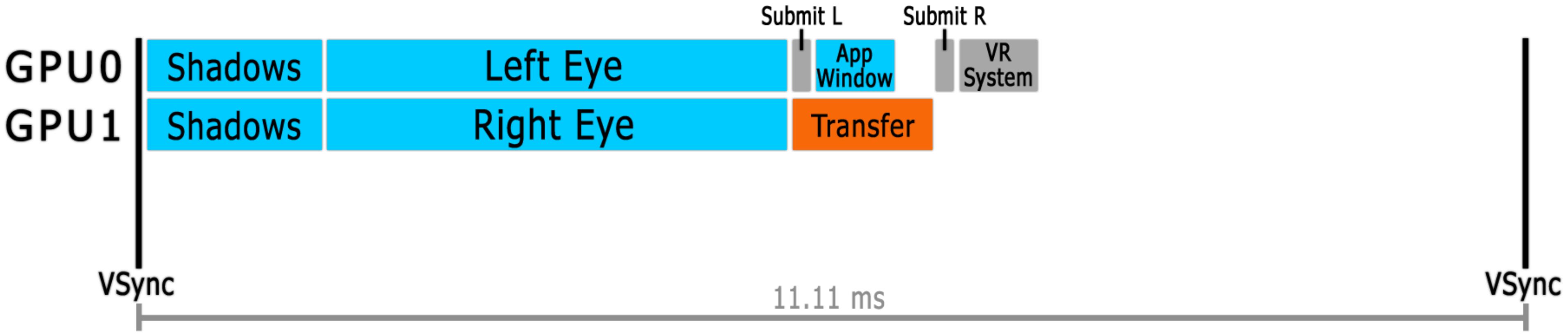
Multi-GPU Affinity APIs

- AMD and NVIDIA have multi-GPU affinity APIs
- Broadcast draw calls across GPUs using affinity mask
- Set different shader constant buffers per-GPU
- Transfer subrects of render targets across GPUs
- Use transfer fences to asynchronously transfer while the destination GPU is still rendering

Multi-GPU – 2 GPUs

- Each GPU renders a single eye
- Both GPUs render shadow buffer
- "Submit Left" and "App Window" executes in the transfer bubble
- 30-35% performance increase

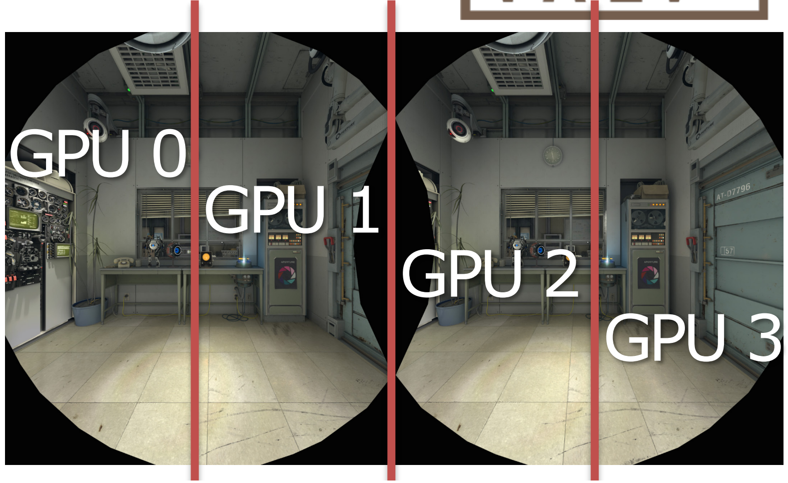


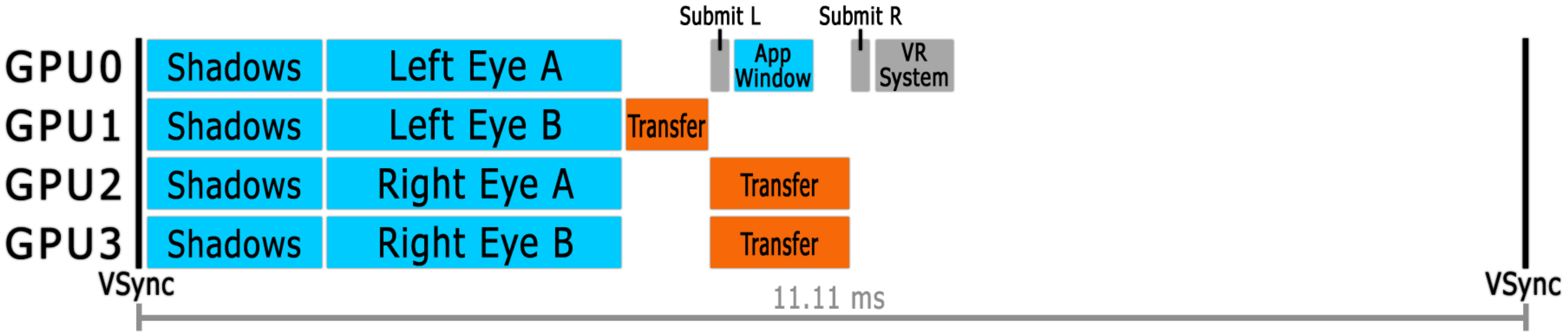


Multi-GPU – 4 GPUs

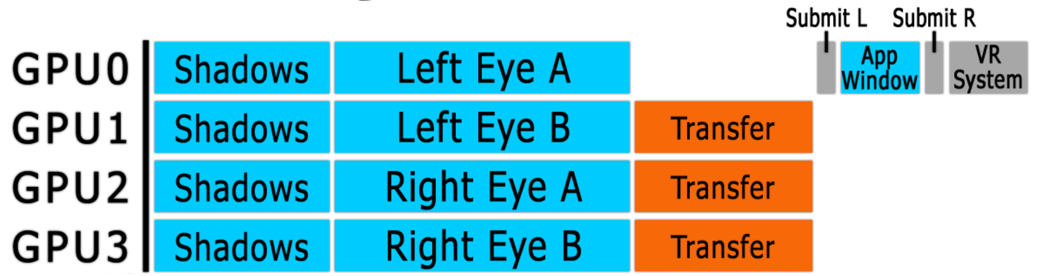
VALVE

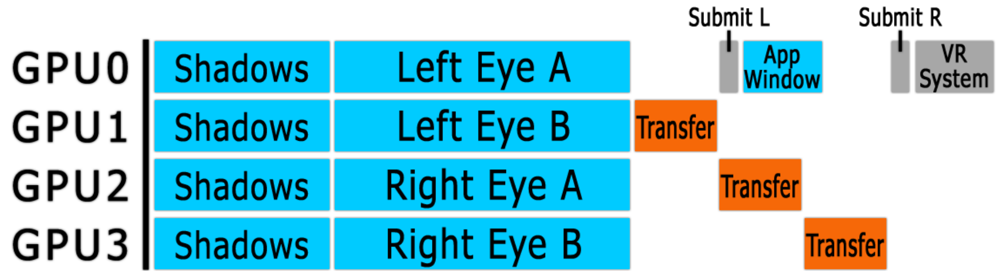
- Each GPU renders half of one eye
- All GPUs render shadow buffer
- PS cost scales, VS cost does not
- Might have high CPU cost in the driver

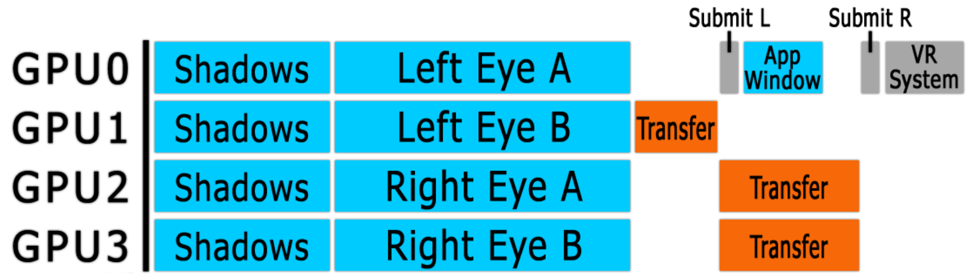




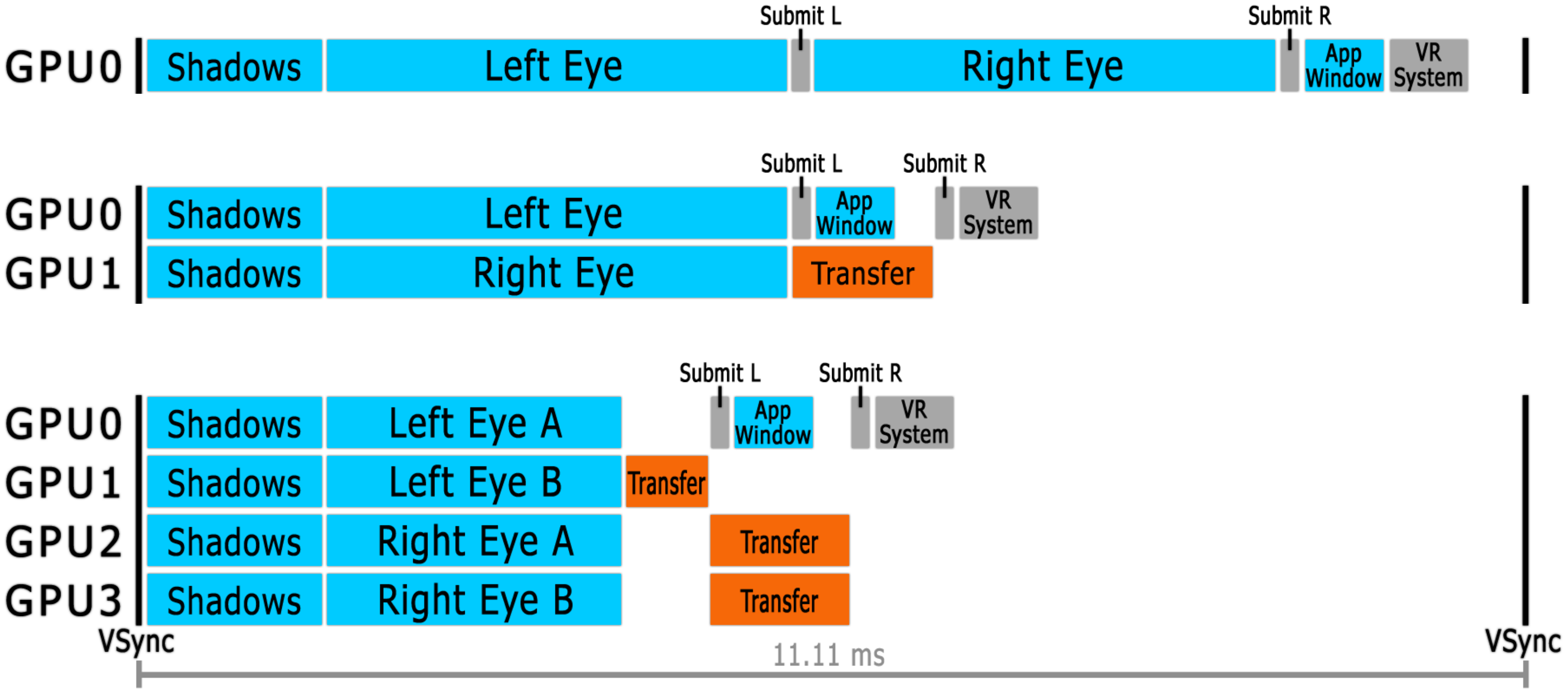
4-GPU Transfer Options



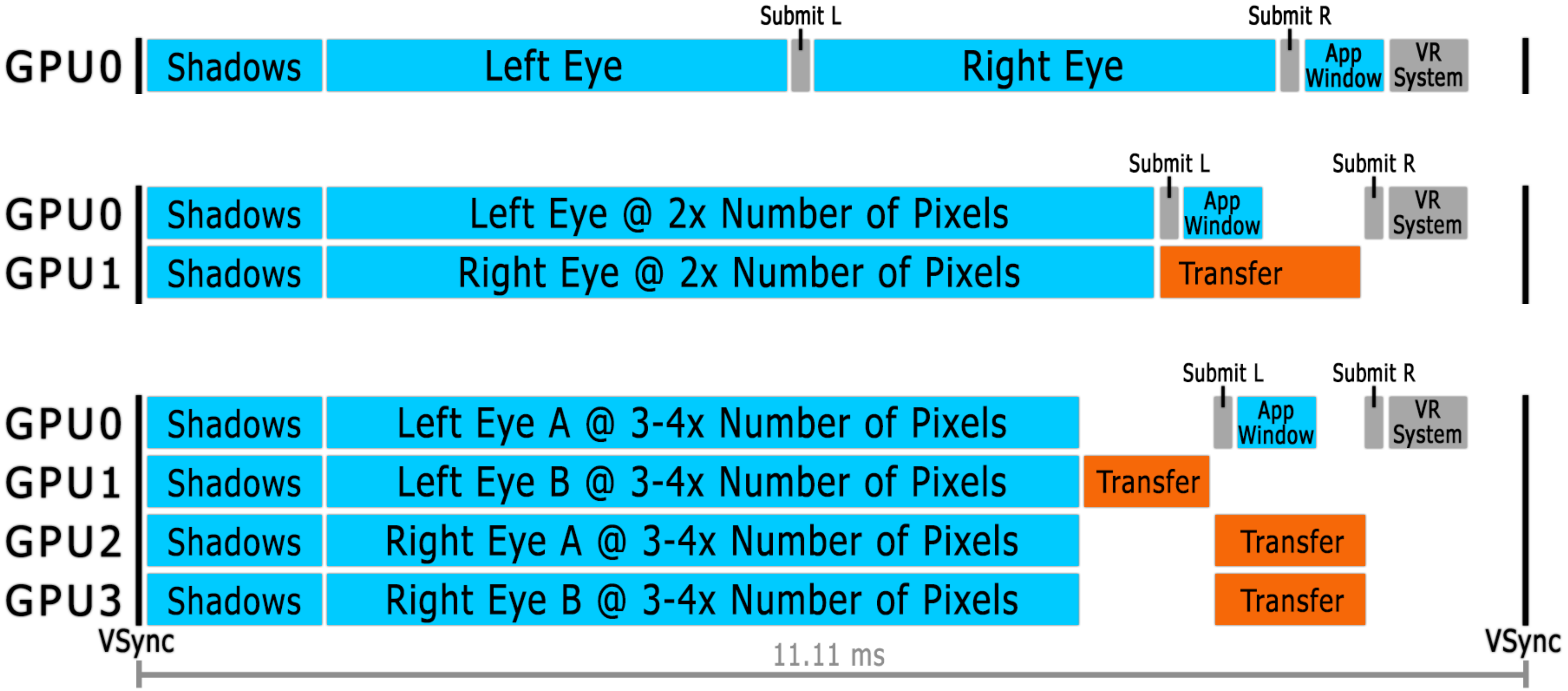




Multi-GPU Performance Summary



Multi-GPU Resolution Scaling



Outline

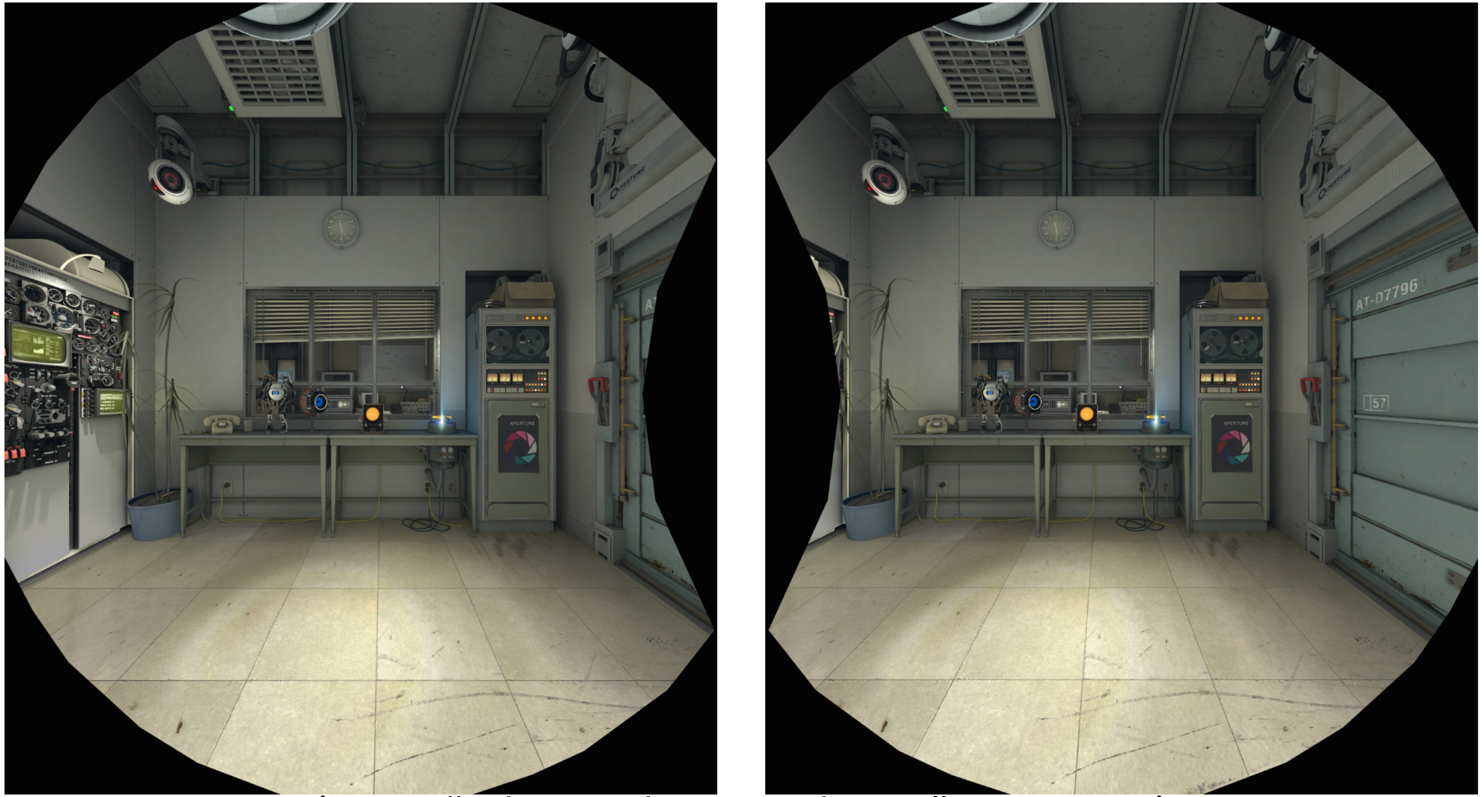
- Multi-GPU for VR
- Fixed Foveated Rendering & Radial Density Masking
- Reprojection
- Adaptive Quality

Projection Matrix vs VR Optics

- Pixel density distribution from the projection matrix is the opposite of what we want
  - Projection matrix: Pixel density per degree increases at the periphery
  - VR optics: Pixel density increases at the center
- We end up over rendering pixels at the periphery

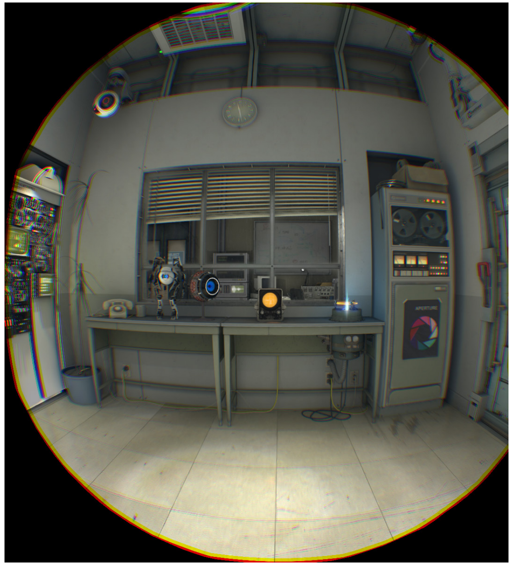
Recap: Over Rendering

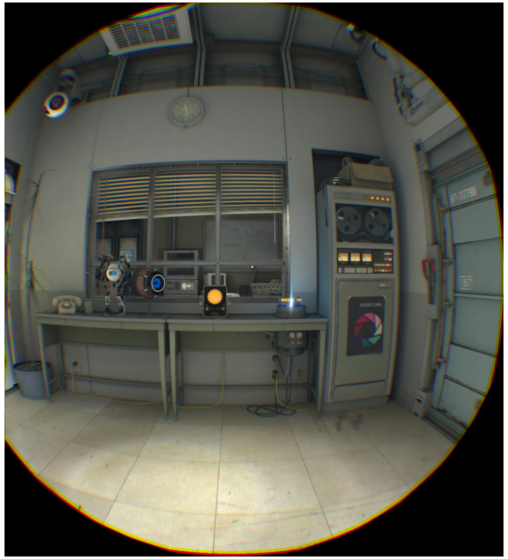
VALVE



(From "Advanced VR Rendering" GDC 2015)

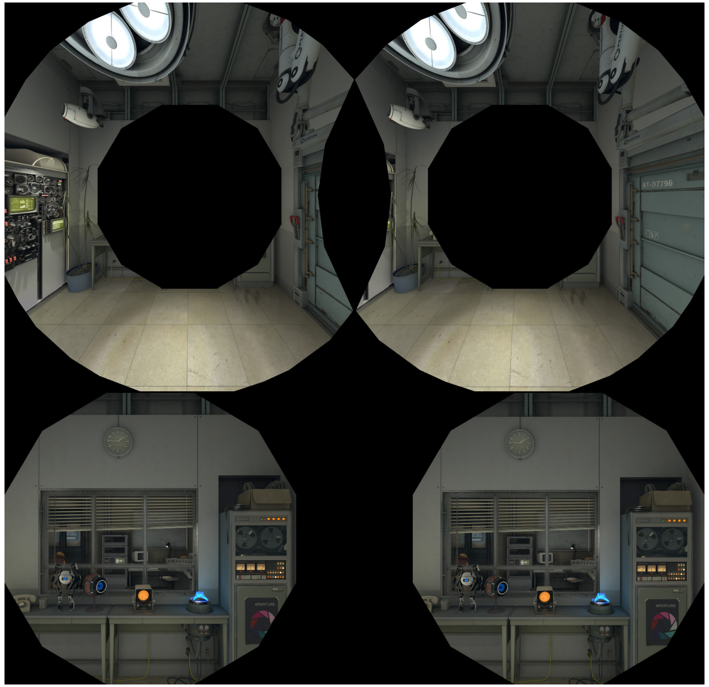
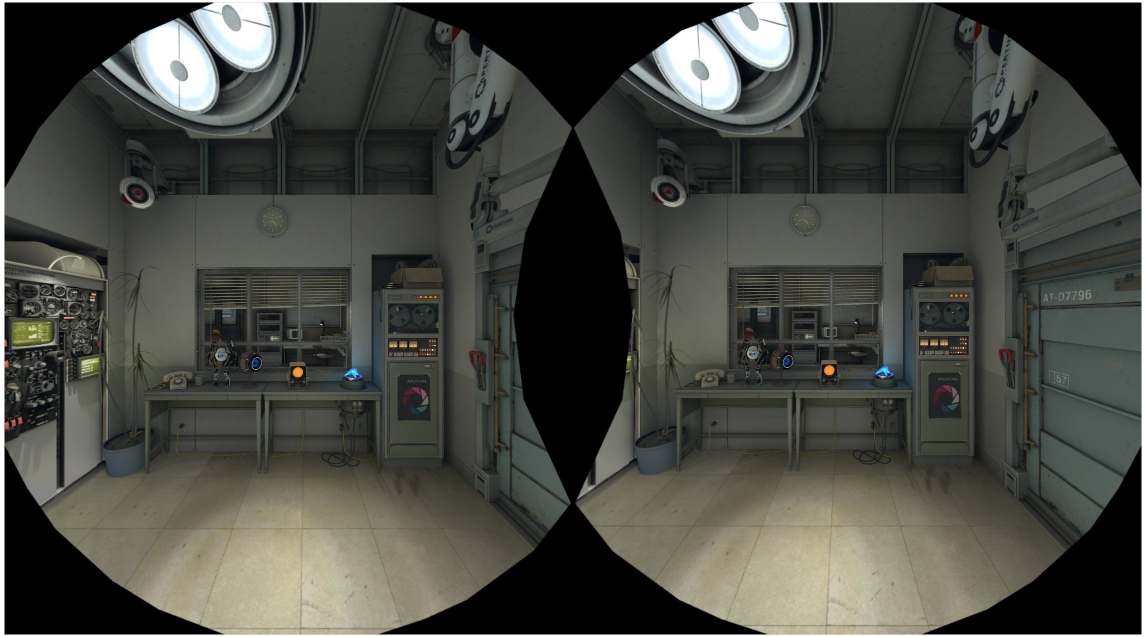
Recap: Over Rendering





(From “Advanced VR Rendering” GDC 2015)

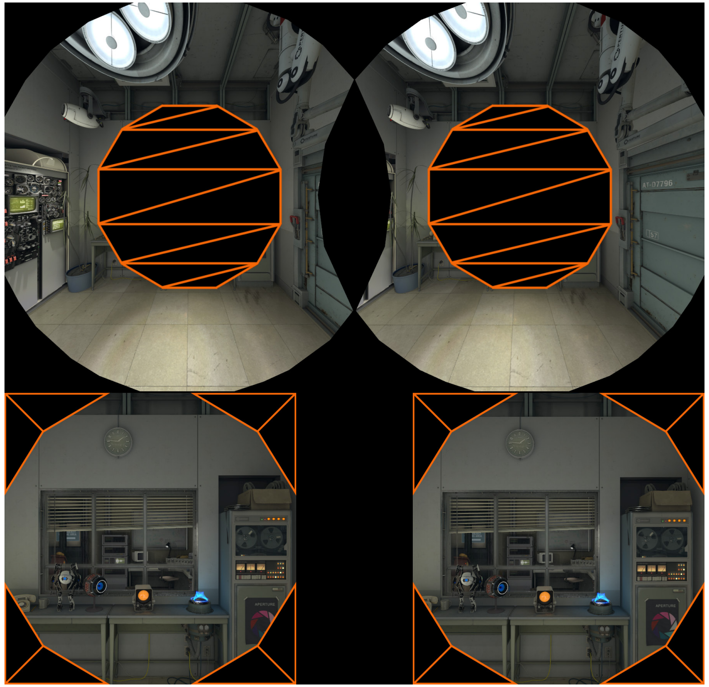
Fixed Foveated Rendering

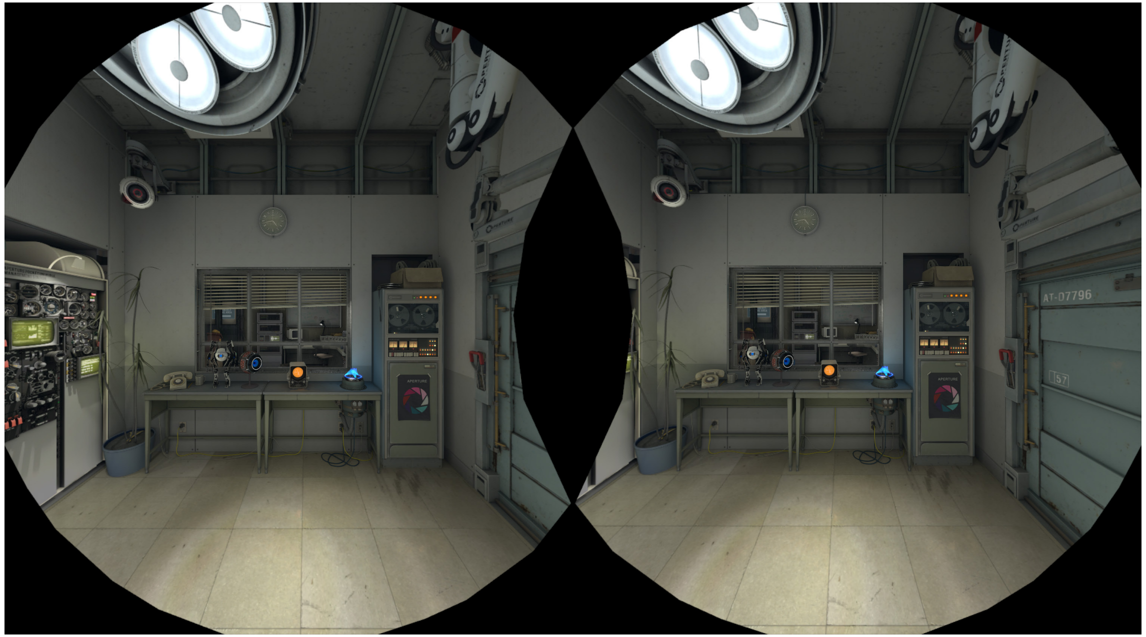



- Multi-GPU opportunities for 2 and 4 GPUs
- Using NVIDIA’s “Multi-Resolution Shading” we gain an additional ~5-10% GPU perf with less CPU overhead (See “*GameWorks VR*”, Nathan Reed, SIGGRAPH 2015)

Fixed Foveated Rendering

VALVE

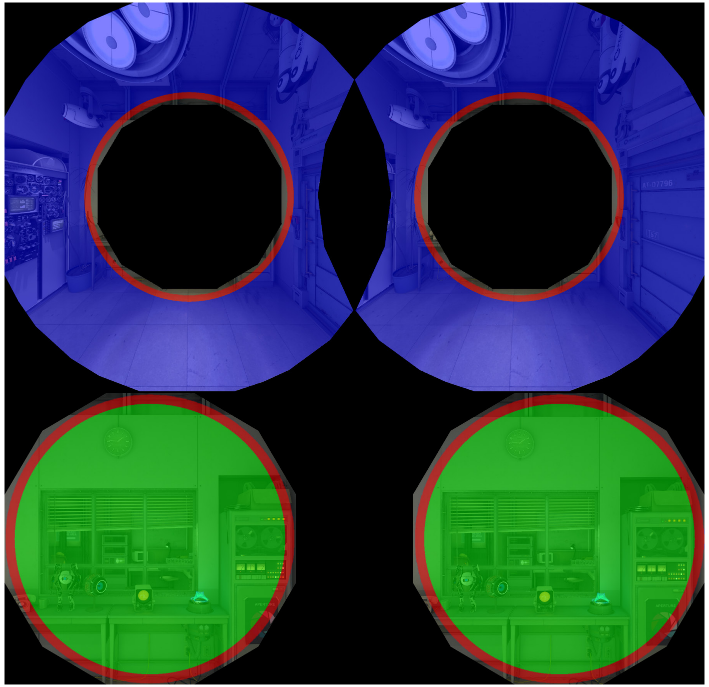


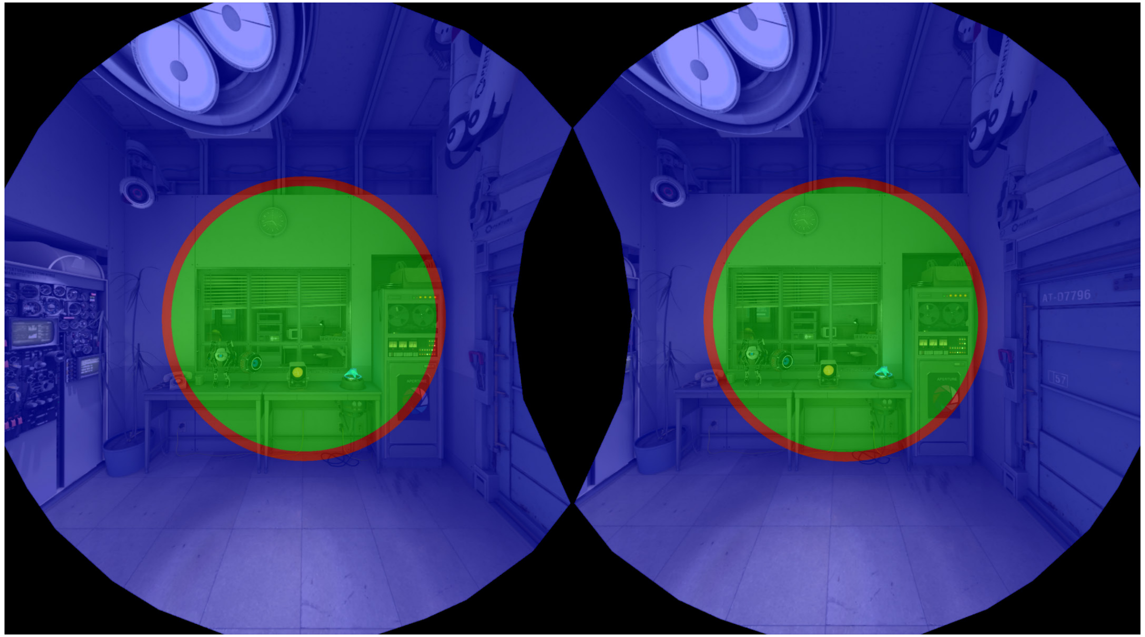


- Multi-GPU opportunities for 2 and 4 GPUs
- Using NVIDIA's "Multi-Resolution Shading" we gain an additional ~5-10% GPU perf with less CPU overhead (See "*GameWorks VR*", Nathan Reed, SIGGRAPH 2015)

Fixed Foveated Rendering

VALVE





- Multi-GPU opportunities for 2 and 4 GPUs
- Using NVIDIA's "Multi-Resolution Shading" we gain an additional ~5-10% GPU perf with less CPU overhead (See "*GameWorks VR*", Nathan Reed, SIGGRAPH 2015)

I’m Bad at Words

Me: “I have this idea to reduce fill rate at the periphery”

Jeep Barnett: “That’s interesting”

<Next day I show a demo>

Jeep: “That’s not even close to what I thought you meant”

Me: “What did you think I meant?”

Jeep: “I thought you would skip rendering every other pixel on the outside”

Me: (Laughing) “That’s not how GPUs work”

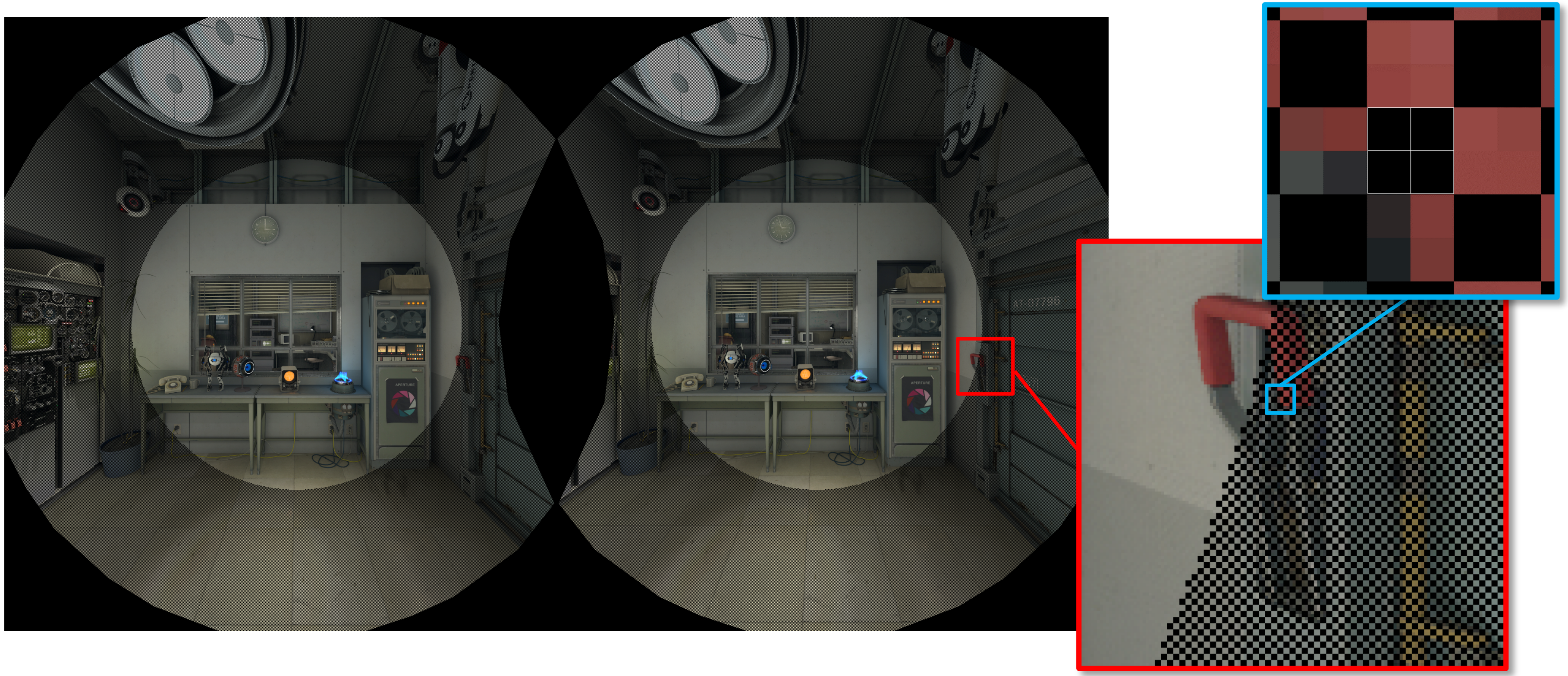
<Later that night>

Me: “Wait a minute…that’s a great idea!”

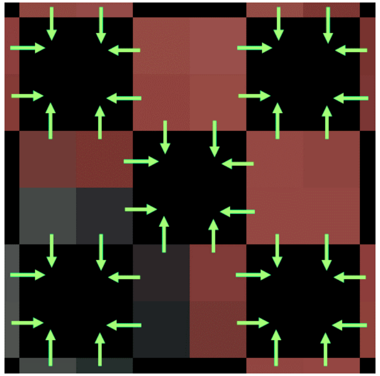
Radial Density Masking

VALVE

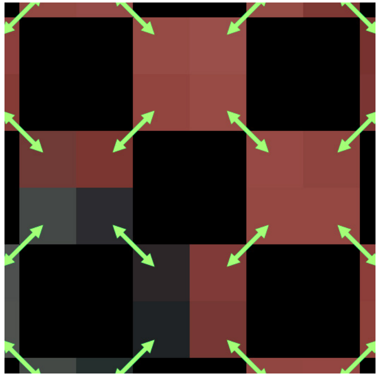
Skip rending a checker pattern of 2x2 pixel quads to match current GPU architectures



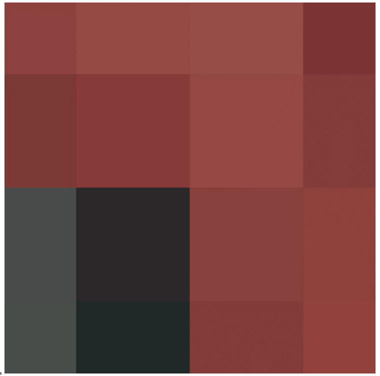
Reconstruction Filter

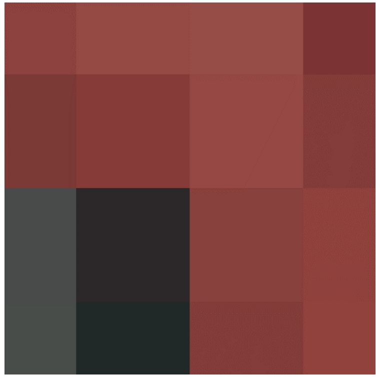

(Average 2 neighbors)

+


(Average across diagonals)

=

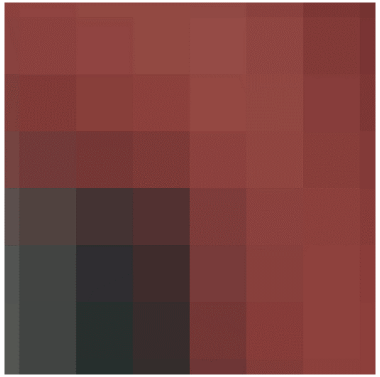




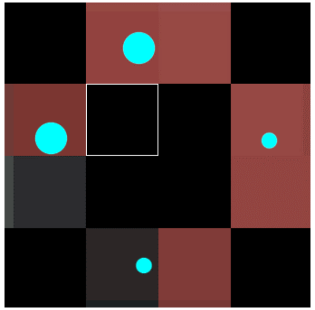
*

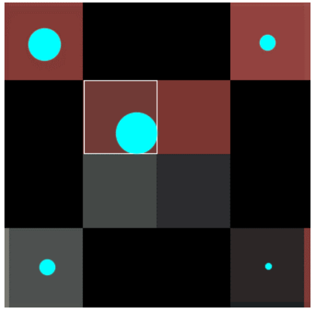
1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

=



Optimized Bilinear Samples


Weights near to far:
0.375, 0.375, 0.125, 0.125


Weights near to far:
0.5, 0.28125, 0.09375, 0.09375, 0.03125

Radial Density Masking

VALVE

1. Clip() 2x2 pixel quads as you render, or fill stencil or depth with a 2x2 checker pattern and render
2. Reconstruction filter

- Saves 5-15% performance in Aperture Robot Repair. You can get higher gains with different content and different shaders. If the overhead of reconstruction and skipping pixels doesn't beat the pixel shader savings of skipped pixel quads, then it's a wash.
- Almost always a big savings on low-end GPUs

Outline

- Multi-GPU for VR
- Fixed Foveated Rendering & Radial Density Masking
- Reprojection
- Adaptive Quality

Dealing With Missed Frames

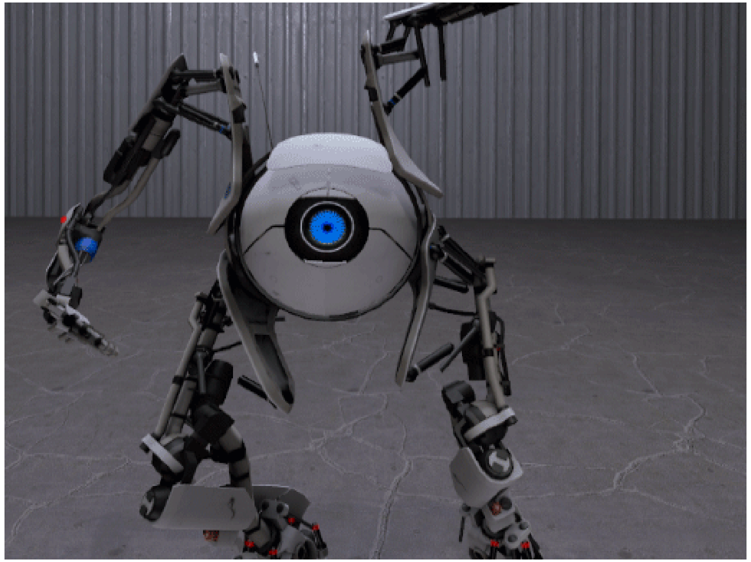
VALVE

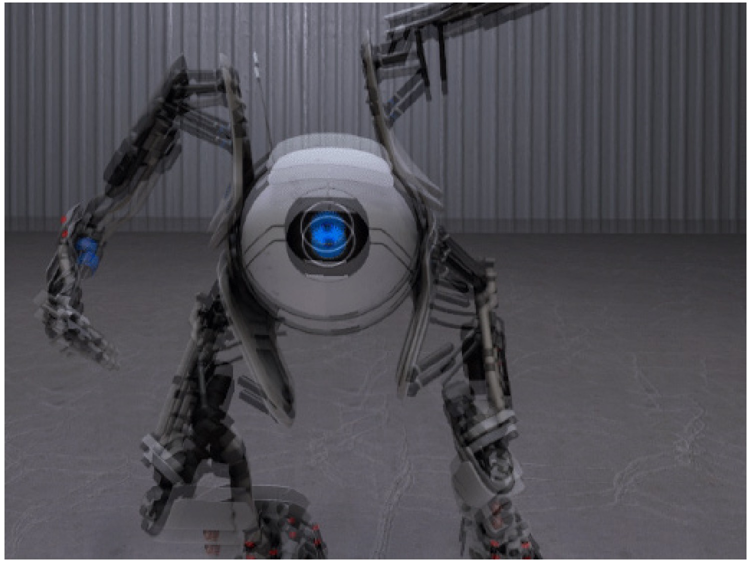
- If your engine is not hitting framerate, the VR system can reuse last frame's rendered images and reproject:
  - Rotation-only reprojection
  - Position & rotation reprojection

- Reprojection to fill in missed frames should be thought of as a last-resort safety net. Please DO NOT rely on reprojection to maintain framerate unless your customer is using a GPU below your application's min spec.

Rotation-Only Reprojection

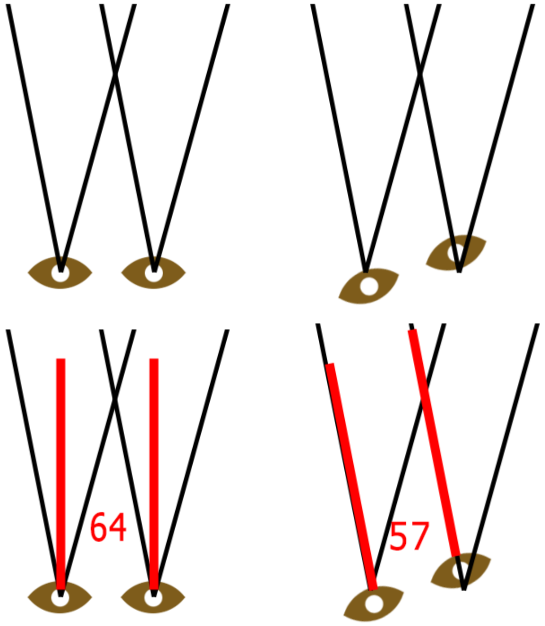
- Judder is caused by camera translation, animation, and objects moved by tracked controllers. Judder appears as two distinct images averaged together.





Rotation-Only Reprojection

- Rotation reprojection is eye-centered, not head centered, so reprojecting from wrong location
- ICD (inter-camera distance) artificially narrows during reprojection depending on amount of rotation
- "A leisurely head turn is in the ballpark of 100 degrees/second" – Michael Abrash, Valve blog, 2013



Rotation-Only Reprojection

- The good:
  - Well-understood algorithm for decades and might improve with modern research
  - It works reasonably well for a single missed frame even with the known side effects

- So…there's a non-trivial set of tradeoffs, but I think it's "good enough" to use as a last-resort safety net for missed frames. It's better than dropping frames.

Positional Reprojection

- Still an unsolved problem that we are very interested in
- You only get one depth in a traditional renderer, so representing translucency is a challenge (Particle systems)
- Depth might be stored in an MSAA depth buffer with color already resolved. This can result in color bleeding.
- Hole-filling algorithms for pixels that aren't represented can cause retinal rivalry. Even with many frames worth of valid stereo pairs, if the user moves vertically by crouching down or standing up, there are gaps that need to be filled.

Asynchronous Reprojection

VALVE

- Ideal safety net
- Requires preemption granularity as good as or better than current generation GPUs
- Current GPUs can generally preempt at draw call boundaries, depending on the GPU
- Not yet a guarantee to always reproject in time for vsync
- Applications need to be aware of preemption granularity:
  - “You can split up the screen into tiles and run the post processing on each tile in a separate draw call. That way, you provide the opportunity for async timewarp to come in and preempt in between those draws if it needs to.” – *“VRDirect”,* Nathan Reed, GDC 2015

Interleaved Reprojection Hint

VALVE

- Older GPUs can't support asynchronous reprojection, so we need an alternative
- OpenVR API has an interleaved reprojection hint where the app can request every-other-frame rotation-only reprojection if the underlying system doesn't support always-on, asynchronous reprojection. App gets ~18ms/frame to render.
- Underlying VR system can also use interleaved reprojection as an auto-enabled safety net when application is below target framerate
- Every-other-frame reprojection is a good tradeoff:
  - "In our experience, ATW should run at a fixed fraction of the game frame rate. For example, at 90Hz refresh rate, we should either hit 90Hz or fall down to the half-rate of 45Hz with ATW. This will result in image doubling, but the relative positions of the double images on the retina will be stable. Rendering at an intermediate rate, such as 65Hz, will result in a constantly changing number and position of the images on the retina, which is a worse artifact." – "*Asynchronous Timewarp Examined*", Michael Antonov, Oculus blog, March, 2015

Outline

- Multi-GPU for VR
- Fixed Foveated Rendering & Radial Density Masking
- Reprojection
- Adaptive Quality

Maintaining Framerate is Hard

- VR is more challenging than traditional games because:
  - The user has fine control over the camera
  - Many interaction models allow users to reconfigure the world
- I gave up on tuning rendering and content to hit 90 fps since users can reconfigure content so easily
- Last year at GDC, we got Robot Repair to hit framerate by tuning the worst 20% of the experience

Adaptive Quality

VALVE

- Stated simply: “Adaptive Quality dynamically changes rendering settings to maintain framerate while maximizing GPU utilization”
  - Goal #1: Reduce the chances of dropping frames and reprojecting
  - Goal #2: Increase quality when there are idle GPU cycles

- Example is the Aperture Robot Repair VR demo running at target framerate on an NVIDIA 680 using two different methods

Adaptive Quality - Benefits

- Lower GPU min spec for your application
- Increased art asset limits – Artists can now make the tradeoff between slightly lower fidelity rendering for higher poly assets or more complex materials
- Don’t need to rely on reprojection to maintain framerate
- Unexpected Benefit: Our apps look better on all hardware

What Settings Are Changed?

- What you can't adjust:
  - Can't toggle visual features like specular
  - Can't toggle shadows
- What you can adjust:
  - Rendering resolution/viewport (aka Dynamic Resolution)
  - MSAA level or anti-aliasing algorithm
  - Fixed Foveated Rendering
  - Radial Density Masking
  - etc.

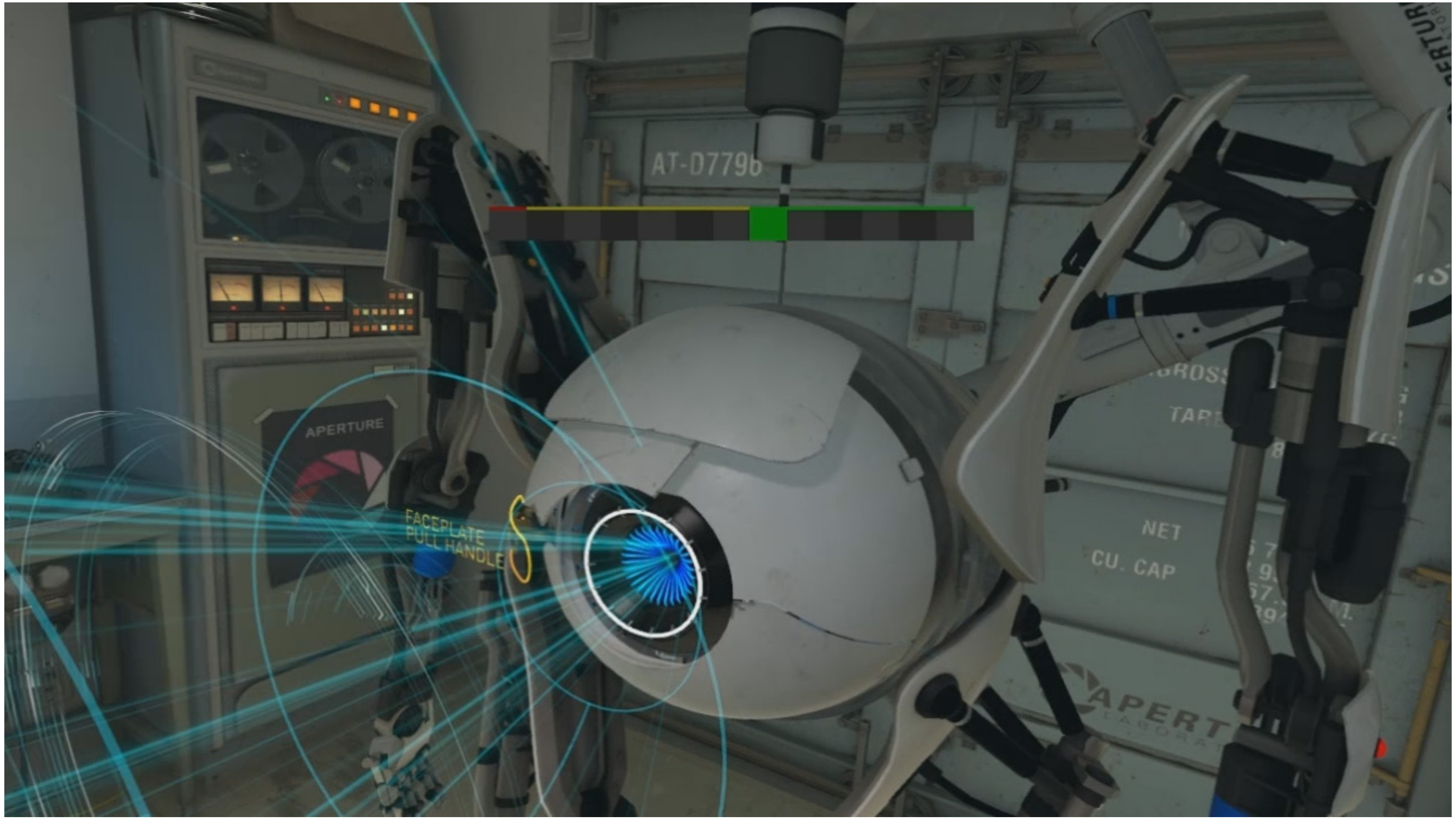
Adaptive Quality Example

Quality Level	MSAA	Resolution Scale	Radial Density Masking	Render Resolution
+6	8x	1.4	-	2116x2352
+5	8x	1.3	-	1965x2184
+4	8x	1.2	-	1814x2016
+3	8x	1.1	-	1663x1848
+2	8x	1.0	-	1512x1680
+1	4x	1.1	-	1663x1848
0	**4x**	**1.0**	**-**	**1512x1680**
-1	4x	0.9	-	1360x1512
-2	4x	0.81	-	1224x1360
-3	4x	0.73	-	1102x1224
-4	4x	0.65	On	992x1102

Default $\longrightarrow$ (row 0)

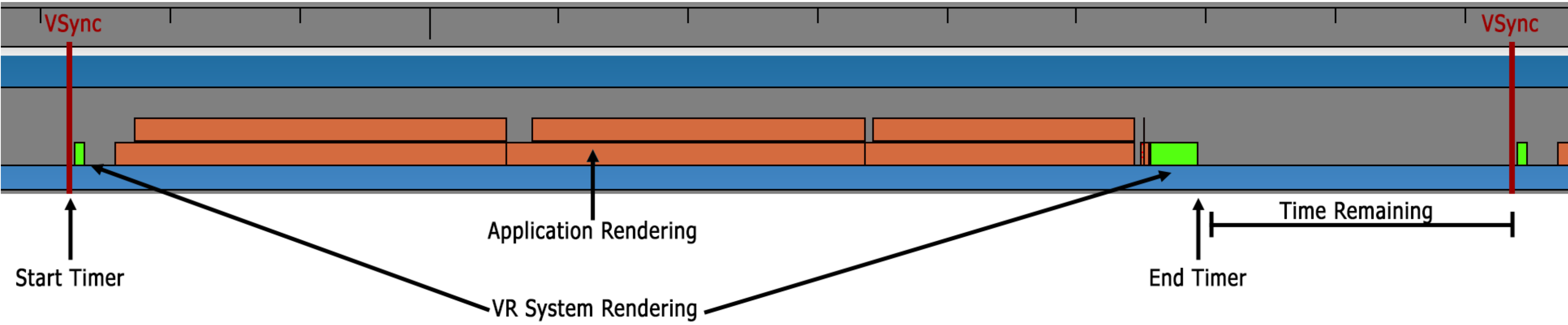
Video of Adaptive Quality Visualization

VALVE



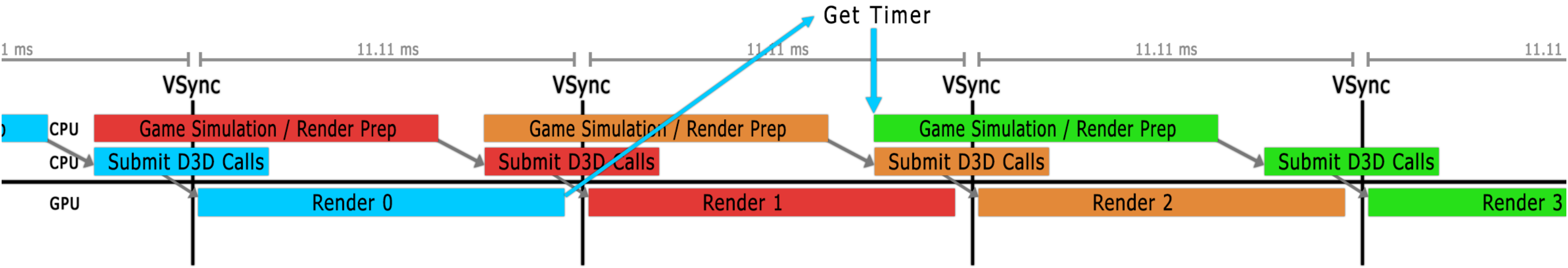
Measuring GPU Workload

- You GPU workload isn't always solid, might have bubbles
- VR system GPU workload is variable: lens distortion, chromatic aberration, chaperone bounds, overlays, etc.
- Get timings from the VR system, not your application. OpenVR, for example, provides a total GPU timer that accounts for all GPU work



GPU Timers - Latency

- Your GPU queries are 1 frame old
- You also have 1 or 2 frames in the queue that can't be modified



Implementation Details – 3 Rules

- Goal: Maintain 70%-90% GPU utilization
- High = 90% of frame (10.0ms)
  - Decrease aggressively: If the last frame finished rendering after the 90% threshold of the GPU frame, drop 2 levels, wait 2 frames
- Low = 70% of frame (7.8ms)
  - Increase conservatively: If the last 3 frames finished below the 70% threshold of the GPU frame, increase 1 level, wait 2 frames
- Prediction = 85% of frame (9.4ms)
  - Use linear extrapolation from last two frames to predict rapid increases
  - If last frame is above the 85% threshold and the linearly extrapolated next frame is above the high threshold (90%), drop 2 levels, wait 2 frames

10% Idle Rule

- The high threshold of 90% leaves 10% of the GPU idle for other processes almost every frame. This is a good thing.

- You need to share the GPU with other processes, even Windows desktop needs a slice of the GPU every few VR frames.

- My mental model of GPU budget changed from last year’s 11.11ms to now 10.0ms per frame, so you almost never starve other processes of GPU cycles.

Adaptive Quality in Aperture Robot Repair

Option A

+6: 8xMSAA, 1.4x res (NVIDIA 980Ti renders here for 95% of the experience)

+5: 8xMSAA, 1.3x res

+4: 8xMSAA, 1.2x res

+3: 8xMSAA, 1.1x res

+2: 8xMSAA, 1.0x res

+1: 4xMSAA, 1.1x res

0: **4xMSAA, 1.0x resolution** (Default) (NVIDIA 970 stays at or above this level)

-1: 4xMSAA, 0.9x res

-2: 4xMSAA, 0.81x res

-3: 4xMSAA, 0.73x res

-4: 4xMSAA, 0.65x res, Radial Density Masking (NVIDIA 680 stays at or above this level)

What About Text?

VALVE

- You don’t actually want to go down to that low resolution scalar of 0.65, because in-game text will be very difficult to read
- Instead:
  - Raise the low end up to about 0.8 resolution scalar
  - If GPU can’t maintain framerate at lowest resolution, enable Interleaved Reprojection Hint (in case asynchronous reprojection isn’t supported)
- For Adaptive Quality, we enable the interleaved reprojection hint when we want to drop below the lowest quality level as the last-resort safety net with rotation-only reprojection

Adaptive Quality in Aperture Robot Repair

VALVE

Option A

+6: 8xMSAA, 1.4x res

+5: 8xMSAA, 1.3x res

+4: 8xMSAA, 1.2x res

+3: 8xMSAA, 1.1x res

+2: 8xMSAA, 1.0x res

+1: 4xMSAA, 1.1x res

0: **4xMSAA, 1.0x resolution** (Default)

-1: 4xMSAA, 0.9x res

-2: 4xMSAA, 0.81x res

-3: 4xMSAA, 0.73x res

-4: 4xMSAA, 0.65x res, Radial Density Masking

Option B – Text-friendly

+6: 8xMSAA, 1.4x res

+5: 8xMSAA, 1.3x res

+4: 8xMSAA, 1.2x res

+3: 8xMSAA, 1.1x res

+2: 8xMSAA, 1.0x res

+1: 4xMSAA, 1.1x res

0: **4xMSAA, 1.0x resolution** (Default)

-1: 4xMSAA, 0.9x res

-2: 4xMSAA, 0.81x res

-3: 4xMSAA, 0.81x res, **Interleaved Reprojection Hint**

Why Max Out at 1.4x Resolution?

VALVE

	Scalar	MSAA	Resolution	GPU Memory 1 Eye = Color + Depth + Resolve	GPU Memory 2 Eyes = Color + Depth + Resolve
	2.0	8x	3024x3360	698 MB	1,396 MB
	2.0	4x	3024x3360	388 MB	776 MB
Aperture →	**1.4**	**8x**	**2116x2352**	**342 MB**	684 MB
	1.2	8x	1814x2016	251 MB	502 MB
	1.0	8x	1512x1680	174 MB	348 MB
Aperture →	**1.1**	**4x**	**1663x1848**	**117 MB**	234 MB
	1.0	4x	1512x1680	97 MB	194 MB
	0.81	4x	1224x1360	64 MB	128 MB

Aperture allocates both a 1.4 8xMSAA and a 1.1 4xMSAA render target per eye for a total of 342 MB + 117 MB = **459 MB per eye (918 MB 2 eyes)**! So we use sequential rendering to share the render target and limit resolution to 1.4x for 4 GB GPUs.

What About 2.0 Resolution Scalar?

VALVE

	Scalar	MSAA	Resolution	GPU Memory 1 Eye = Color + Depth + Resolve	GPU Memory 2 Eyes = Color + Depth + Resolve
Aperture →	**2.0**	**8x**	**3024x3360**	**698 MB**	1,396 MB
	2.0	4x	3024x3360	388 MB	776 MB
	1.4	8x	2116x2352	342 MB	684 MB
	1.2	8x	1814x2016	251 MB	502 MB
	1.0	8x	1512x1680	174 MB	348 MB
Aperture →	**1.1**	**4x**	**1663x1848**	**117 MB**	234 MB
	1.0	4x	1512x1680	97 MB	194 MB
	0.81	4x	1224x1360	64 MB	128 MB

For a 2.0 resolution scalar, we require **698 MB + 117 MB = 815 MB** per eye.

Valve’s Unity Rendering Plugin

- Valve is using a custom rendering plugin in Unity for The Lab
- The Valve VR Rendering Plugin will be free on the Unity Asset Store soon with full source code
- The plugin is a single-pass forward renderer (because we want 4xMSAA and 8xMSAA) supporting up to 18 dynamic shadowing lights and Adaptive Quality
- Special thanks to Unity devs Peter Kuhn, Scott Flynn, Joachim Ante, and Rich Geldreich for adding Adaptive Quality hooks to Unity 5.4.0b9 that shipped one week ago!

Decoupling CPU and GPU Performance

- Make your render thread autonomous
- If CPU isn't ready with a new frame, don't reproject! Instead, render thread resubmits last frame's GPU workload with updated HMD poses and minimal Adaptive Quality support of dynamic resolution
- To solve animation judder, feed your render thread two animation frames you can interpolate between to keep animation updating
- But, non-trivial animation prediction is a hard problem
- Then you can plan to run your CPU at 1/2 or 1/3 GPU framerate to do more complex simulation or run on lower end CPUs

Summary

- Multi-GPU support should be in all VR engines (at least 2-GPUs)
- Fixed Foveated Rendering and Radial Density Masking are solutions that help counteract the optics vs projection matrix battle
- Adaptive Quality scales fidelity up and down while leaving 10% of the GPU available for other processes. Do not rely on reprojection to hit framerate on your min spec!
- Valve VR Rendering Plugin for Unity will ship free soon
- Think about how your engine can decouple CPU and GPU performance with resubmission on your render thread

Thank You!

Alex Vlachos, Valve

Alex@ValveSoftware.com