# GRAPHICS PROCESSING REQUIREMENTS FOR ENABLING IMMERSIVE VR

By David Kanter
Sponsored by AMD

Over the last three decades, graphics technology has evolved and revolutionized the way that people interact with their computers, cars, smartphones, and other devices. Driven primarily by advances in computer game technology, performance has improved by many orders of magnitude, programmability and graphical quality have flourished, and as costs have come down, computer graphics have become entrenched in modern life.

While computer graphics have evolved dramatically, up until recently computer display technology has advanced following a more incremental trajectory. Over the past 20 to 30 years, while screen resolution, the number of colors available per pixel, and other features have increased and improved, consumer graphics have almost exclusively been displayed via a simple two-dimensional display – whether on a PC monitor, a TV, or a smart phone.

Fast forward to today — the vast improvements in graphics technology, as well as related technology developments in smart phones and consumer electronics, mean that virtual reality (VR) is now emerging as a viable possibility. The promise of VR — a fully immersive experience — is incredibly alluring to the computer and film industries, developers, and consumers alike. The appeal is so strong that as with video accelerators in the early 1990's, there are now a number of different companies developing VR solutions, all seeking to fundamentally redefine the way that people experience computing.

Achieving true immersion for virtual or augmented reality (AR, which is effectively compositing a virtual world with the real world) and creating a compelling user experience is no trivial task. The challenges are entirely different than those facing previous consumer technologies: the audio-visual experience is central to immersion. Unlike graphics intended for games, movies or television, looking brilliant on a static display that is measured in mere inches is insufficient; VR typically requires a head-mounted display (HMD) that freely moves the display in tandem with the head. The goal of VR/AR is true immersion — nothing less than creating an illusion so real that it convinces the human brain, the world's finest computer. The audio-visual experience is therefore fundamentally rooted in the human audio-visual system, and requires an entirely new approach to the problem.

Speaking to the visual aspect of virtual reality, making the transition from 2D display graphics to an immersive display poses several significant challenges. First, the resolution requirements must be reframed in the context of the human visual system, which demands dramatically more pixels than today's standard display. At the same time, this higher resolution must be displayed at lower latencies to preserve the continuous illusion of reality. Not only is VR performance more demanding than standard 2D graphics, but the inputs are also more complex since VR systems rely on head and gesture tracking. To meet these challenges and successfully deliver immersive experiences, developers must be allowed to use graphics software and hardware in new and ingenious ways. AMD's LiquidVR™ SDK is an ideal platform for VR

development, including many features which simplify and optimize VR development, and also unlock unique hardware features such as asynchronous shading.

**Resolution**

Educated consumers often think about display quality in terms of pixels per inch (PPI), which is an excellent metric for a simple two-dimensional display such as a monitor or a smartphone that occupies a relatively fixed point in space. For virtual reality however, the objective is to fully surround an individual and delight their visual system with an immersive experience. As a result, when working with the human eye and visual system, it is more appropriate to think about the field of view (FoV), which is measured in degrees or arc-minutes (1/60$^{th}$ of a degree) and about pixels per degree (PPD)[1], rather than in pixels per inch.

As Figure 1 shows, the human visual system is amazingly complex: each eye has a horizontal field of view of 160° and a 175° vertical field of view[2]. The two eyes work together for stereoscopic depth perception over a 120°-wide and 135°-high FoV, and can move roughly 90 degrees in as little as ⅒ of a second during saccades (small motions as the eye changes focus points)[3]. Not only is the instantaneous FoV quite wide, but after accounting for head rotations, humans can see in a full 360° horizontal arc.


**Figure 1: Human field of view for virtual reality compared to PC monitors**

[1] Pixels per degree is the number of pixels across each degree of the field of view, which accounts for the distance from the display, whereas pixels per inch does not. For example, a display that has a 100° field of view and 7,000 pixels is 70 pixels per degree.

[2] http://webvision.med.utah.edu/book/part-xiii-facts-and-figures-concerning-the-human-retina/

[3] Visual Search 2: Proceedings of the 2$^{nd}$ International Conference on Visual Search, p270; Optican 1995

One of the more remarkable aspects of the human visual system is that it is both wide (in terms of FoV), but also incredibly deep in terms of its ability to perceive fine detail. The eye senses colors using cones that are attuned to blue, green, and red/yellow. The most sensitive part of the eye, known as the fovea, contains most of the cones, and has a resolution of about 1 arc-minute, which translates into 60 pixels-per-degree (PPD)[4]. Other portions of the eye, especially close to the periphery, are significantly less sensitive to fine details, but are more sensitive to latency and detecting fast changes.

| *For a single eye* | **PPD** | **Horizontal FoV°** | **Horizontel Kpixels** | **Vertical FoV°** | **Vertical Kpixels** | **Total Mpixels** |
|---|---|---|---|---|---|---|
| **Full FoV Resolution** | 60 | 160 | 9.6 | 175 | 10.5 | 100.8 |
| **Stereo FoV Resolution** | 60 | 120 | 7.2 | 135 | 8.1 | 58.32 |

**Chart 1: Virtual reality rendering requirements**

What does this translate into for VR? As Chart 1 shows, an immersive display with 60 pixels per degree requires an incredible 9.6K horizontal and 10.5K vertical pixels, or 100.8 MPixels for each eye! Delivering this resolution without any changes to current rendering methods is clearly an aspirational goal, but would provide the most immersive and convincing VR experience possible. Given the pace of Moore's Law and the trajectory of semiconductor technology over the next decade, optimization and smart design choices are necessary. As incredibly sophisticated as the human eye is, there are some relatively simple opportunities for optimization. For example, given the geometry of the human eye, humans can only see in full stereo over roughly 120° horizontally and 135° vertically[5]. Limiting the FoV accordingly would reduce the display requirement by almost 50% to 7.2K x 8.1K or 58.3 MPixels per eye. While certainly a difficult goal, delivering this resolution to each eye should be achievable within the next decade.

**Latency**

One of the keys to a successful VR experience is matching the sensory input of the virtual world with human sensory expectations. The user-perceived latency is often described as 'motion to photon', i.e., the length of time between an input (e.g., changing the head position) and the moment when a full frame of pixels reflecting the associated changes is displayed. While the resolution for virtual reality can be scaled back considerably to accommodate economic and technical realities, latency is an entirely different story: the latency between the GPU and the display is a critical component of a very tightly-coupled and highly-sensitive system.

[4] Johnson, David. Human Factors and Displays for VR Graphics.

[5] http://webvision.med.utah.edu/book/part-xiii-facts-and-figures-concerning-the-human-retina/

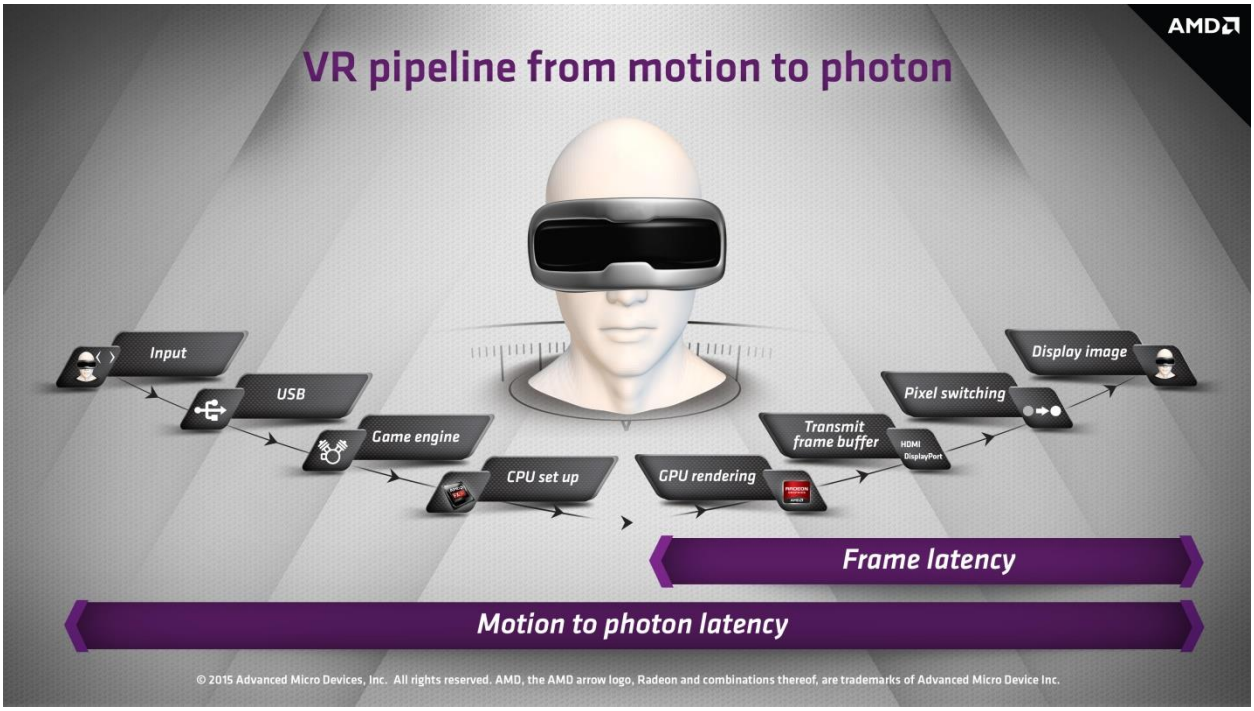


**Figure 2: Motion-to-photon and frame latency in VR systems**

In some traditional graphics workloads, low latency can be essential to high-quality rendering, but in many cases, high resolution or visual effects take priority. For example, most movies operate at 24 frames per second, and a game is considerable 'playable' at 30 frames per second. In contrast, low latency is absolutely essential for true immersion and arguably the most important performance metric for VR. High frame latency causes the virtual world to become decoupled from the motions and orientation of the user (e.g., dropping a frame, known as judder) - not only breaking the immersive experience, but easily causing disorientation and nausea (also known as simulation sickness).

To avoid simulation sickness in the general population, the motion-to-photon latency should never exceed 20 milliseconds (ms)[6]. As Figure 2 shows, the frame latency (i.e., GPU rendering through to image display) is about half the motion-to-photon latency; the rest of the time is needed to receive and process gesture inputs and calculate changes in the virtual world. To provide a pleasant and immersive experience though, a frame latency of 5ms is achievable for current VR systems (equivalent to a 10ms motion-to-photon latency). However, even at 5ms, this represents a dramatic shift for the graphics world given that frame latency only recently became a high design priority (compared to frame rates).

**Comparison between VR and Gaming**

Given the intense graphical and computational demands of VR applications, it is only natural to look at the rendering technology used in today's most taxing consumer graphics applications: PC gaming. While

[6] http://blogs.valvesoftware.com/abrash/latency-the-sine-qua-non-of-ar-and-vr/

there are many similarities between PC games and VR applications and games, there are also fundamental differences, as highlighted in Chart 2.

As previously mentioned, traditional graphics APIs and supporting hardware have evolved over 20 years and have been carefully optimized for throughput, often measured in frames per second (FPS). In contrast, VR systems are acutely sensitive to frame latency and subtle timing variations – much more akin to real-time computing than to a 'throughput at all cost' approach. As a result, the target latency from input-to-display update in VR systems is easily three to 10 times lower than for today's standard visualization applications[7]. This is not only an issue for the GPU, which has less time to render a frame, but also for the display device itself. Modern commodity LCD displays typically operate at a 60 Hz frame rate, i.e., 16.6ms, whereas many VR systems are capable of 10ms (or 100 Hz) and would further benefit from 5ms (200 Hz) or even lower. As designing low-latency displays (i.e., with a refresh rate below 10ms) has not been as crucial to the industry to date, VR will help drive this technology forward.

| | **Gaming PC** | **Stereo VR** |
|---|---|---|
| **Display** | 2D LCD panel | Stereo HMD |
| **Field of View (FoV)** | 24-30" diagonal | 120° horizontally<br>135° vertically |
| **Number of pixels** | 2-4 Mpixels | 116 Mpixels (for two eyes) |
| **Frame latency** | 16-33ms | 5ms |
| **Input** | Mouse and keyboard | Multi-sensor, gesture tracking |

**Chart 2: Differences between PC gaming and stereo VR**

The inputs and outputs are also dramatically different between the VR and standard visualization pipelines. PCs, consoles, and mobile devices all use fixed-position 2D displays with an implicit expectation that the user is staring at the display. Initial VR systems are based on HMDs, which use special optics to present the user with a wide FoV while simultaneously tracking changes in body and head position and maintaining the HMD weight, power, and thermals at tolerable levels. While HMDs are becoming the standard technology for VR, new approaches may use entirely different display technologies. Future systems may employ projected VR displays and highly-realistic positional audio as well as natural input with haptic and other, more advanced forms of sensory feedback.

The inputs for VR are also vastly more complicated than for a standard PC. The latter relies on a mouse and keyboard, which are fairly straightforward to handle. On the other hand, virtual reality systems use a variety of sensors to track information about the user and the inputs, which are then fed back into the virtual world. Each virtual reality system functions slightly differently but head tracking via a combination of magnetometers, accelerometers, gyros, and external position tracking hardware (e.g., Valve's

[7] http://blogs.valvesoftware.com/abrash/latency-the-sine-qua-non-of-ar-and-vr/. A motion-to-photon latency of 10ms would provide a pleasant and immersive VR experience. Today's games have a motion-to-photon latency as low as 30ms for simpler games, and up to 100ms or more with triple buffering, etc.

Lighthouse or Oculus's IR Camera) is becoming a de facto minimum standard for virtual reality systems connected to desktop PCs. Future systems may even track the center of focus of the eyes precisely, e.g., for foveated rendering. Many VR systems also use gesture input, which requires further tracking of the user's hands. All of these inputs must be received (typically over USB) and then processed by the game engine. Head tracking data is used to predict and determine the viewing angles used for graphics rendering, as well as positional audio, while other inputs may cause additional state changes to the virtual world.

Given these differences, the near-term focus for the VR ecosystem is on delivering the illusion of presence today by effectively enabling today's VR software and hardware, and ramping up the angular resolution and graphics quality to rival that of the latest display technology. Two of the best metrics for image quality are measuring the single precision FLOP count per pixel and the bytes of memory bandwidth per pixel for a single frame. Together, these two numbers give a very rough approximation of how much computation and data can be used to make each pixel look good and account for both resolution and frame latency.

| *For a single eye* | **Hor. Kpixels** | **Vert. Kpixels** | **Total Mpixels** | **Frame Latency (ms)** | **Stereo Mpixels/ sec** | **Bytes/ pixel** | **FLOP count/ pixel** |
|---|---|---|---|---|---|---|---|
| **Oculus Rift**[8] | 1.08 | 1.2 | 1.296 | 5.56 | 466.6 | 702.33 | 12,022.74 |
| **Project Morpheus**[9] | 0.96 | 1.08 | 1.0368 | 4.17 | 497.7 | 658.44 | 11,252.57 |
| **DK2**[10] | 0.96 | 1.08 | 1.0368 | 6.67 | 311.0 | 1053.50 | 18,004.12 |
| **HTC Re ViVe**[11] | 1.2 | 1.08 | 1.296 | 5.56 | 466.6 | 702.33 | 12,022.74 |
| **Average Gaming PC**[12] | 1.92 | 1.08 | 2.0736 | 33 | 125.7 | 2,607.41 | 44,560.19 |
| **High-end Gaming PC**[13] | 2.56 | 1.6 | 4.096 | 16 | 248.2 | 1,320.00 | 22,558.59 |
| **Stereo FoV @ 60 PPD**[14] | 7.2 | 8.1 | 58.32 | 5 | 23,328.0 | 14.05 | 240.05 |

**Chart 3: Differences between PC gaming and selected VR systems**[15]

Chart 3 shows a comparison of the graphics requirements and image quality metrics for Oculus' DK2 and the Valve/HTC Re Vive, alongside that of conventional PC gaming and of a hypothetical 60 PPD VR system. The PC gaming system specifications are based on data from the Steam Hardware Survey, which indicates

[8] https://developer.oculus.com/blog/powering-the-rift/
[9] http://blog.eu.playstation.com/2015/03/03/gdc-2015-project-morpheus-update/
[10] https://developer.oculus.com/documentation/intro-vr/latest/reference/faq-product/
[11] http://www.htcvr.com/
[12] Average Gaming PC system specifications: 24" PC monitor, 1920x1080 resolution, 91.8 PPI, 30 Hz display refresh rate
[13] High-end Gaming PC system specifications: 30" PC monitor, 2560x1600 resolution, 100.6 PPI, 60 Hz display refresh rate
[14] Stereo VR target specifications: 120° horizontal FoV, 135° vertical FoV, 60 PPD, 200 Hz display refresh rate
[15] HMDs require a separate display for each eye whereas PCs use a single monitor for both eyes.

that 96% of users have resolutions of 1920x1080 or below, and 99.6% of users have resolutions of 2560x1600 or below[16].

As seen in Chart 3, the actual number of pixels rendered per second for VR systems at a good frame latency is roughly four times higher than that of the average PC display (1080P resolution), and twice that of a high-end PC display. Problematically, this disparity in pixel count directly translates into a two to four times lower bandwidth and FLOPs per pixel, so VR displays simply cannot use the same complex shaders for rendering while maintaining consistent frame latency.

**SDKs Bridge the Gap from Gaming to VR**

Today's graphics vendors and VR developers must rely on intelligent software and hardware optimizations to enable consistent VR performance, high resolution and immersive image quality. To that end, vendors have released software development kits that optimize the VR pipeline and take full advantage of the underlying graphics and display hardware. AMD's LiquidVR SDK is one of the most advanced of its kind, and is valuable for current and future VR rendering techniques.

**Direct-to-Display**

The majority of operating systems today do not have special support for the HMDs typically used for VR. As an example, Windows® 7, the most popular operating system according to Steam's March Survey, was released prior to the first generation of prototypes from Oculus VR or Valve/HTC. As these operating systems do not recognize the special requirements associated with VR, they attempt to manage HMDs in the same way as a standard monitor. The OS might attempt to extend the desktop to a VR display or apply power management strategies which are not necessarily compatible with the VR environment. As a result, displaying a monoscopic 2D desktop on a 3D stereoscopic display can be incredibly jarring and uncomfortable.

LiquidVR helps developers to make the transition from PC-style display technologies to the immersive HMDs that are common in VR. The SDK removes the responsibility for interfacing with the VR display from the OS, providing two major benefits. First, LiquidVR is designed to natively drive VR displays. It thereby can enable a more seamless plug and play experience on host OSes that were not designed for VR (e.g., Windows® 7 and more recently, Windows® 8) and eliminates the need for shim drivers to redirect output, which would otherwise create another layer of complexity. Second and more importantly, removing the OS from the critical path enables much tighter control over rendering and display. In conventional DirectX® or OpenGL rendering, the display output is typically double or triple buffered, and the GPU renders to the last frame in the buffer. The LiquidVR SDK enables direct rendering to the front buffer, significantly reducing latency.

Rendering direct-to-display, also known as front buffer rendering, in conjunction with asynchronous shaders and latest data latch, is crucial for using the latest possible head tracking data to warp the display 'just in time' and deliver the lowest possible perceived motion-to-photon latency.

[16] Steam 2015 March hardware survey

**Affinity Multi-GPU**

The traditional approaches to leveraging multiple GPUs used by AMD CrossFire™ and NVIDIA SLI™ technologies, such as Alternate Frame Rendering (AFR), have been successful at increasing frame rates of conventional PC games. However, AFR is not suitable for VR as it is designed today, does not improve latency, and in some cases, can actually increase it. On the other hand, because VR is stereoscopic, there exists a very natural opportunity to easily and efficiently use one or more GPUs *per eye* in order to decrease latency while simultaneously driving higher resolution displays and providing better overall image quality. As Figure 2 shows, the LiquidVR SDK can intelligently partition the overall rendering pipeline to minimize CPU burden and avoid replication between the two GPUs.

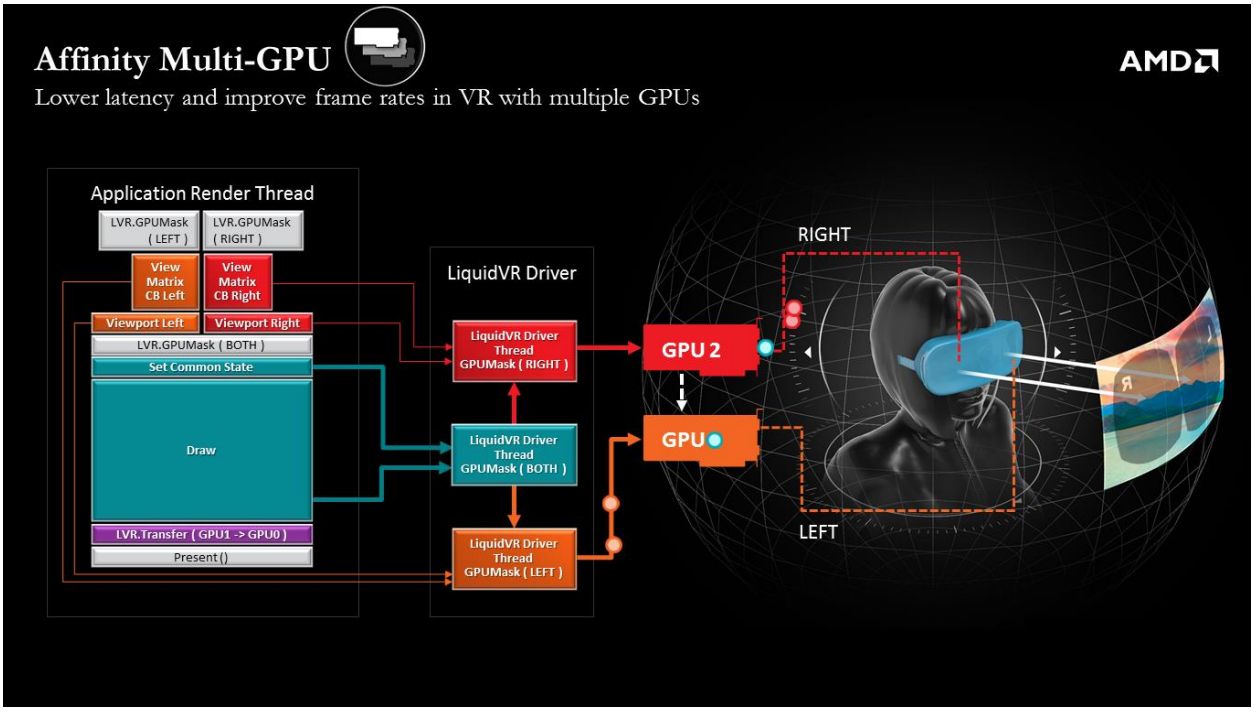


**Figure 2: Lower latency and improved frame rates with multiple GPUs**

Using LiquidVR's Affinity Multi-GPU API, an application can create a single command stream (e.g., state changes and draw calls) which can be simultaneously broadcast to any combination of GPUs in the system. The application can modify a simple 'GPU Affinity' mask that controls which GPUs will receive any given API call. The majority of the rendering seen by each eye is normally very similar, so the same draw calls can usually be submitted to both eyes. In many cases, only the viewport and the view matrix differ between the two eyes.

For a scenario in which one GPU is assigned to each eye, the application sends most commands to the driver once instead of twice, and the commands can be processed in parallel. After rendering is completed, the results will typically be transferred from the slave GPU across a PCIe® bus to the master GPU, which will then send the combined output to the actual display over HDMI or DisplayPort. In addition, this data transfer can occur asynchronously since AMD GPUs have independent DMA engines.

By carefully partitioning the work when using two GPUs to accelerate stereo rendering , Affinity Multi-GPU can cut the latency nearly in half and double the bandwidth and computational resources, and reduce CPU latency and improve CPU throughput (e.g., draw calls per second). This technique can be extended to four or more GPUs, by partitioning each frame within the eye (e.g., two GPUs render portions of the left eye frame; two other GPUs render of the right eye frame).

**Latest Data Latch**

One shortcoming of the traditional graphics pipeline is that the GPU is fundamentally treated as an offload engine for the CPU, rather than as a peer in the system. Data (e.g., constants) is pushed from the CPU to the GPU when draw calls are inserted into the command buffer; it can be many milliseconds or even seconds before the data is actually consumed by the GPU. This delay is a huge challenge for latency-sensitive applications like VR. Latest Data Latch is a technique designed to address this issue by enabling the GPU to consume updated data 'just-in-time' rather than receiving older data from the CPU, which is a crucial advantage for VR and other applications that require very low latency. This is especially important for moving sensor input data into the rendering pipeline, which is one of the major changes in VR rendering.

The most obvious way to render a virtual reality scene is to first determine the orientation of the viewer and then render the scene according to that perspective. However, this approach can have uncomfortable results for the viewer; by the time the rendering is finished, the position and orientation which were used for rendering are outdated — creating a mismatch between the human sensory expectations and VR experience.

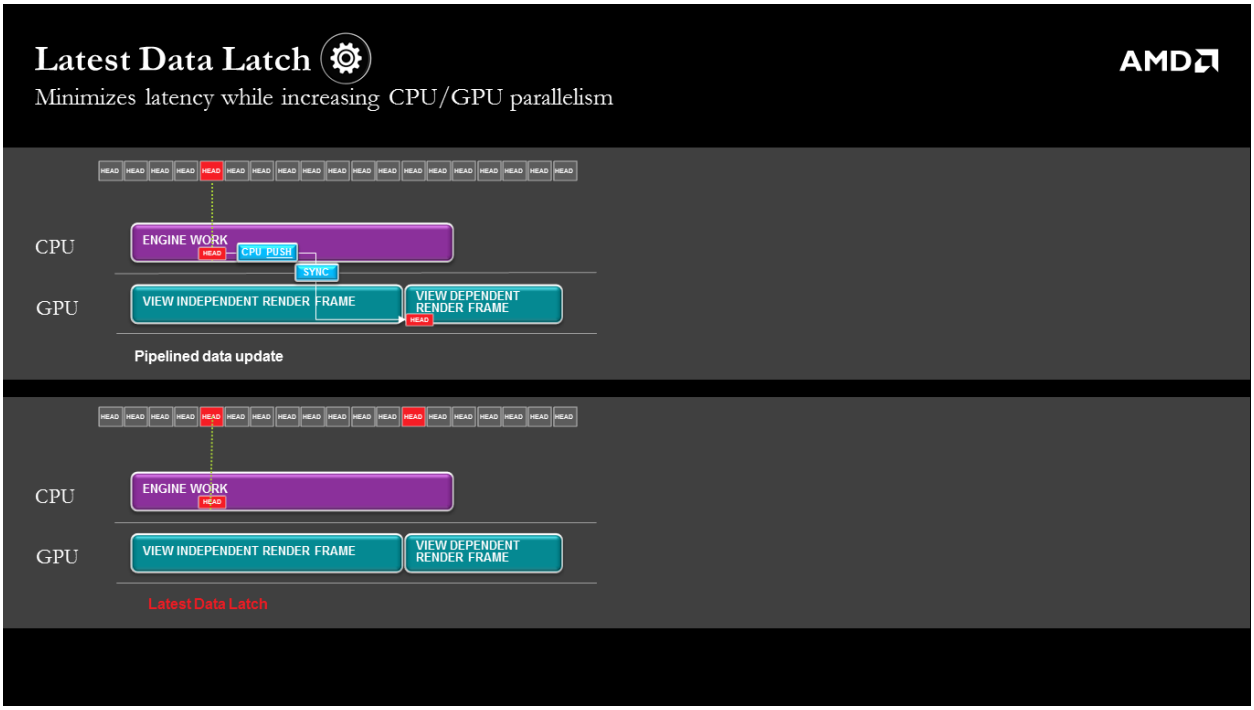


**Figure 3: Latest Data Latch minimizes latency while increasing CPU/GPU parallelism**

As Figure 3 shows, with latest data latching, the application simply injects a copy operation into the GPU command stream, and then the GPU copies the target data when the command is executed rather than receiving stale data. The GPU can then use the most recent orientation data followed by post-processing to 'time warp' a rendered frame. The CPU continuously samples the orientation (e.g., at 1 KHz or higher rates) into a circular buffer with a head pointer to the latest data. The application uses latest data latch to instruct the GPU to copy the head pointer. When the command is executed on the GPU, the head pointer is copied into local memory. At this point, GPU shaders can use the head pointer to sample the latest orientation data from the CPU's buffer without expensive synchronization primitives.

**Asynchronous Shaders for simultaneous computing**

Asynchronous shading is an incredibly powerful feature that increases performance by more efficiently using the GPU hardware while simultaneously reducing latency. In the context of VR, it is absolutely essential to reducing motion-to-photon latency, and is also beneficial for conventional rendering.

AMD's GCN architecture has several separate types of queues that generate work for the GPU: a graphics command processor, a number of asynchronous compute engines, and two copy engines. The graphics command processor dispatches both graphics (hull, vertex, geometry, domain and pixel) and DirectX/OpenGL compute shaders to the cores in the GPU's shader array. The asynchronous compute engines specialize in scheduling compute workloads and synchronization with other portions of the chip. Lastly, the copy engines are responsible for transferring data to and from the graphics memory (e.g., DMAs from main memory to graphics memory).

AMD's GCN architecture can simultaneously schedule and execute work from all three of the above: graphics shaders, compute workloads, and data copies. This significantly improves the real-time characteristics and latency of scheduling heterogeneous compute and graphics workloads, and also has the potential to increase the overall performance of the GPU for mixed workloads, which are particularly common in VR.

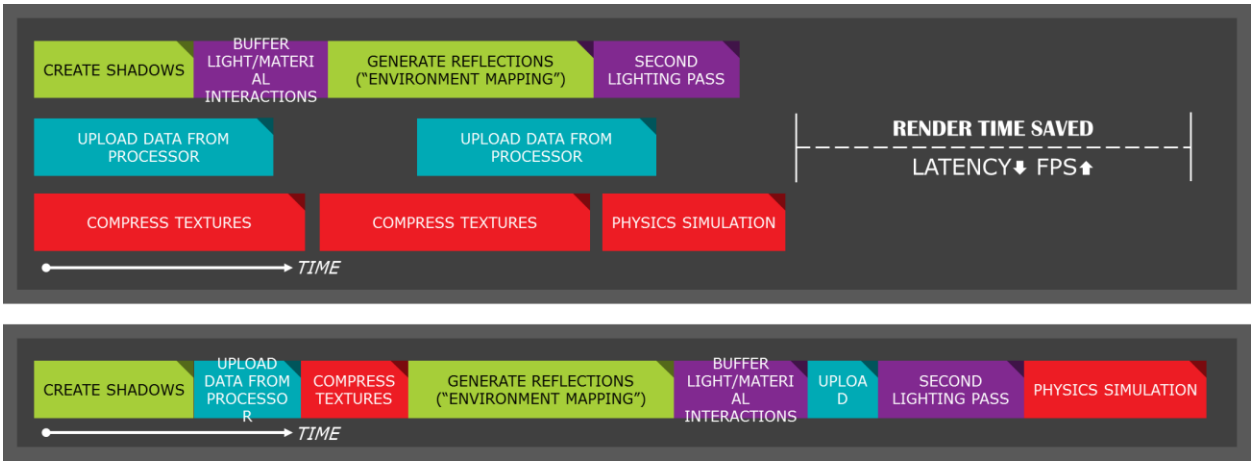


**Figure 4: Asynchronous shading reduces latency by enabling simultaneous graphics shaders, compute shaders, and data copy operations for standard graphics and VR**

For example, VR systems use shaders to eliminate lens distortion in HMDs. Most GPU architectures cannot simultaneously execute compute and graphics shaders and only context switch at draw call boundaries. The scene is first rendered using graphics shaders, then the GPU idles during a high latency context switch before the distortion shader can begin. This idling wastes GPU cycles and adds latency to the overall pipeline. In contrast, AMD's GCN architecture can overlap the execution of the distortion shader with the graphics shaders from the next frame and eliminate the pipeline bubbles caused by context switching. This significantly improves latency and throughput by more efficiently using GPU resources: performance benefits of up to 46% have been measured for VR rendering[17].

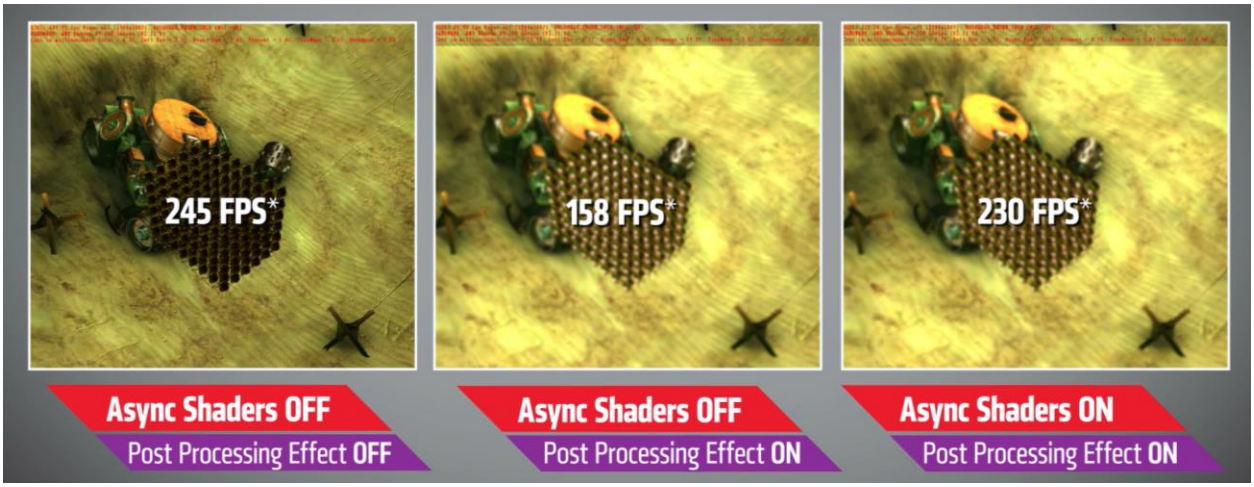


**Figure 5: Async Shaders can significantly improve performance**

As another example, the post-processing time warp discussed above also depends on concurrent compute and graphics shaders. Ideally, the 'time warp' is a just-in-time process that is tied to the frame refresh. Without asynchronous compute, however, context switch latency can be too high and unpredictable; there is a huge risk that the frame will miss the refresh boundary and then cause discomfort or simulation sickness.

On the other hand, applications programmed to utilize GCN-based GPUs powered by AMD's LiquidVR technology can immediately start the 'time warp' process just prior to the vertical sync, and extend this process into the display scan-out time, running completely asynchronously to the next frame (which can start its graphics rendering simultaneously). The end result is that throughput and parallelism are increased, latency is reduced, and judder is eliminated, thereby ensuring a fully immersive experience.

[17] System Specifications: CPU: AMD FX 8350, Memory: 16GB DDR3 1600 MHz, Motherboard: 990 FX, GPU: AMD R9 290X 4GB, OS: Windows 7 Enterprise 64-bit Driver: 15.10.1006, Demo Running: AMD Internal Application – Asynchronous Compute

**Conclusion**

Virtual reality is one of the most exciting developments in computer graphics because it fundamentally redefines how we interact with computers, from the input through to the display output. As with any change of such a great magnitude, the technical challenges are myriad. The industry must simultaneously increase resolution, decrease latency, and incorporate entirely new inputs to create a seamless and immersive experience. As each of these metrics change to improve the experience, the performance required from the GPU increases accordingly; double the resolution and the GPU performance should double as well.

The performance requirements for VR will challenge modern developers and hardware vendors alike; a simple extension of today's brute force rendering techniques will not suffice. To unlock the potential of VR, the industry must enable the entire VR ecosystem to experiment with new rendering techniques and new hardware features that ultimately enhance the VR experience and deliver an immersive experience to the consumer.

For example, recent work by Microsoft Research demonstrated that foveated rendering could reduce the resolution requirements by 100X for a 70° display by decreasing the resolution for portions of the screen that are further away from the user's gaze[18]. This is a perfect example of research that cuts across multiple areas: human optics and perception, display technology, and rendering techniques, and fundamentally changes the graphics paradigm to deliver a wholly different experience. As with any evolving technology, there are tremendous opportunities — but everything starts with the right combination of hardware and software. AMD's recently released LiquidVR™ SDK is a first significant step toward meeting these goals.

[18] Guenter, et al. Foveated 3D Graphics.